Generative Adversarial Networks

Deep Learning (DSE316/616)

Vinod K Kurmi
Assistant Professor, DSE

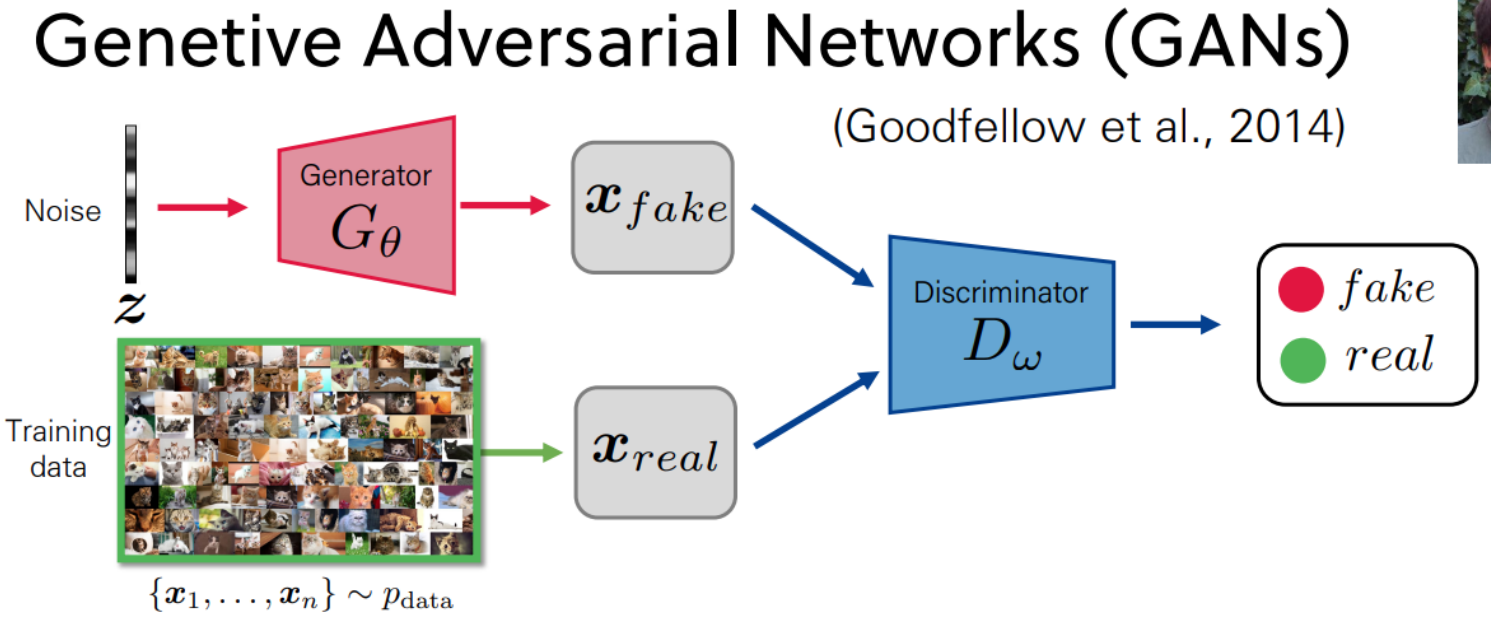
Indian Institute of Science Education and Research Bhopal

Oct 31, 2022

Disclaimer

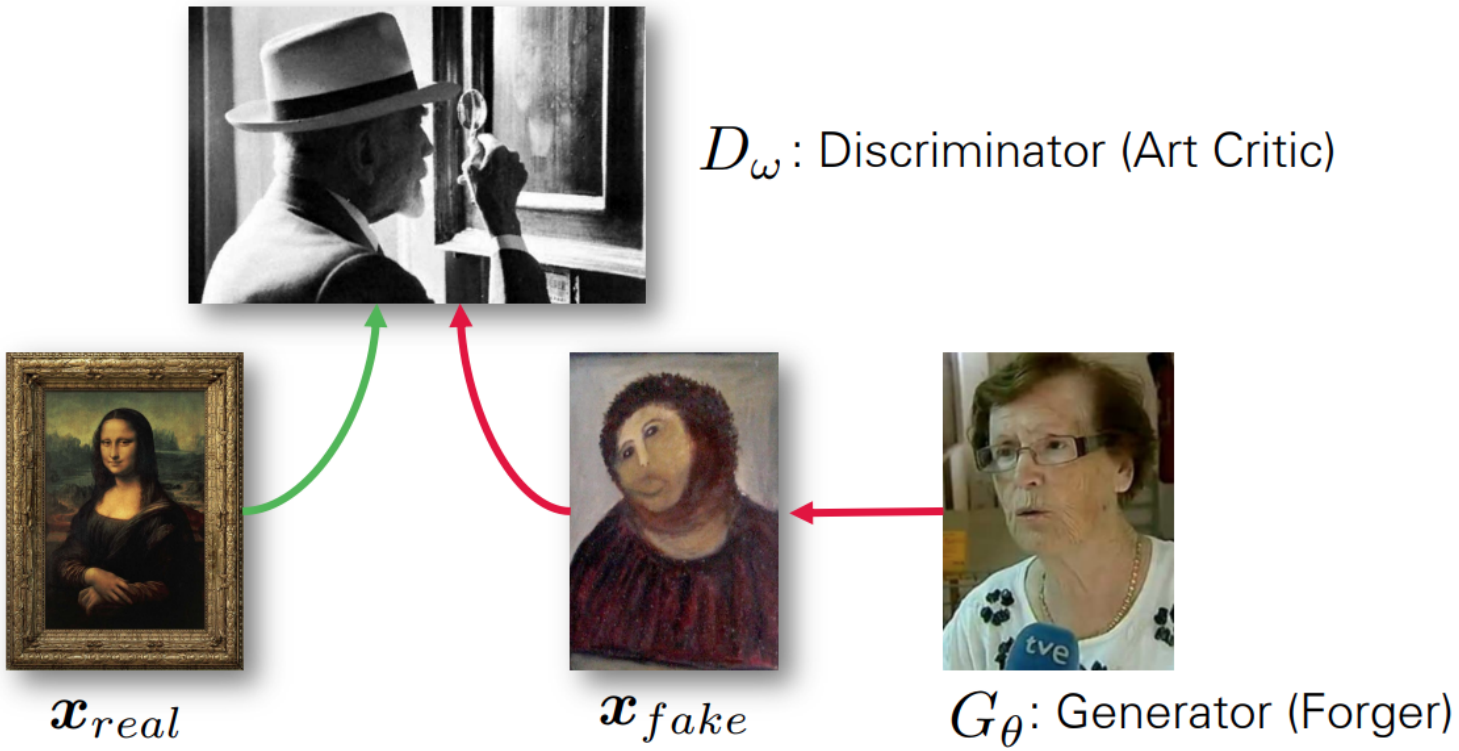
- Much of the material and slides for this lecture were borrowed from
  - Bernhard Schölkopf's MLSS 2017 lecture,
  - Tommi Jaakkola's 6.867 class,
  - CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University
  - Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class
  - Hongsheng Li's ELEG5491 class
  - Tsz-Chiu Au slides
  - Mitesh Khapra Class notes

Genetive Adversarial Networks (GANs)



- A game between a generator $G_\theta(\boldsymbol{z})$ and a discriminator $D_\omega(\boldsymbol{x})$
  - Generator tries to fool discriminator (i.e. generate realistic samples)
  - Discriminator tries to distinguish fake from real samples

Intuition behind GANs



GAN Training: Minimax Game

$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\log D_{\omega}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} \left[\log \left(1 - D_{\omega}(G_{\theta}(\boldsymbol{z}))\right)\right]$$

Real data

Noise vector used to generate data

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log \left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

Cross-entropy loss for binary classification

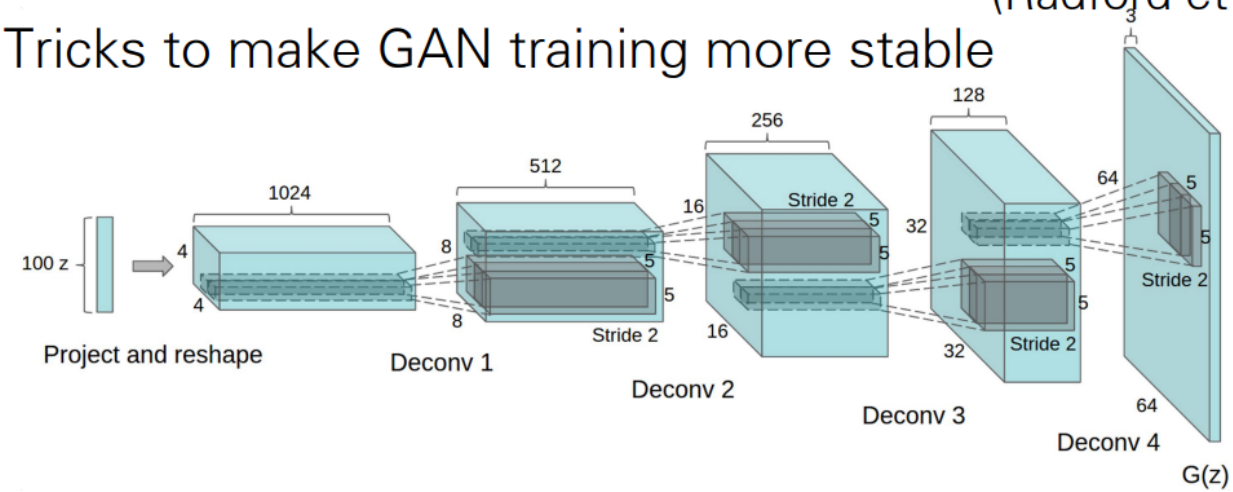
$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D\left(G(\boldsymbol{z})\right)$$

Generator maximizes the log-probability of the discriminator being mistaken

Deep Convolutional GANs (DCGAN)

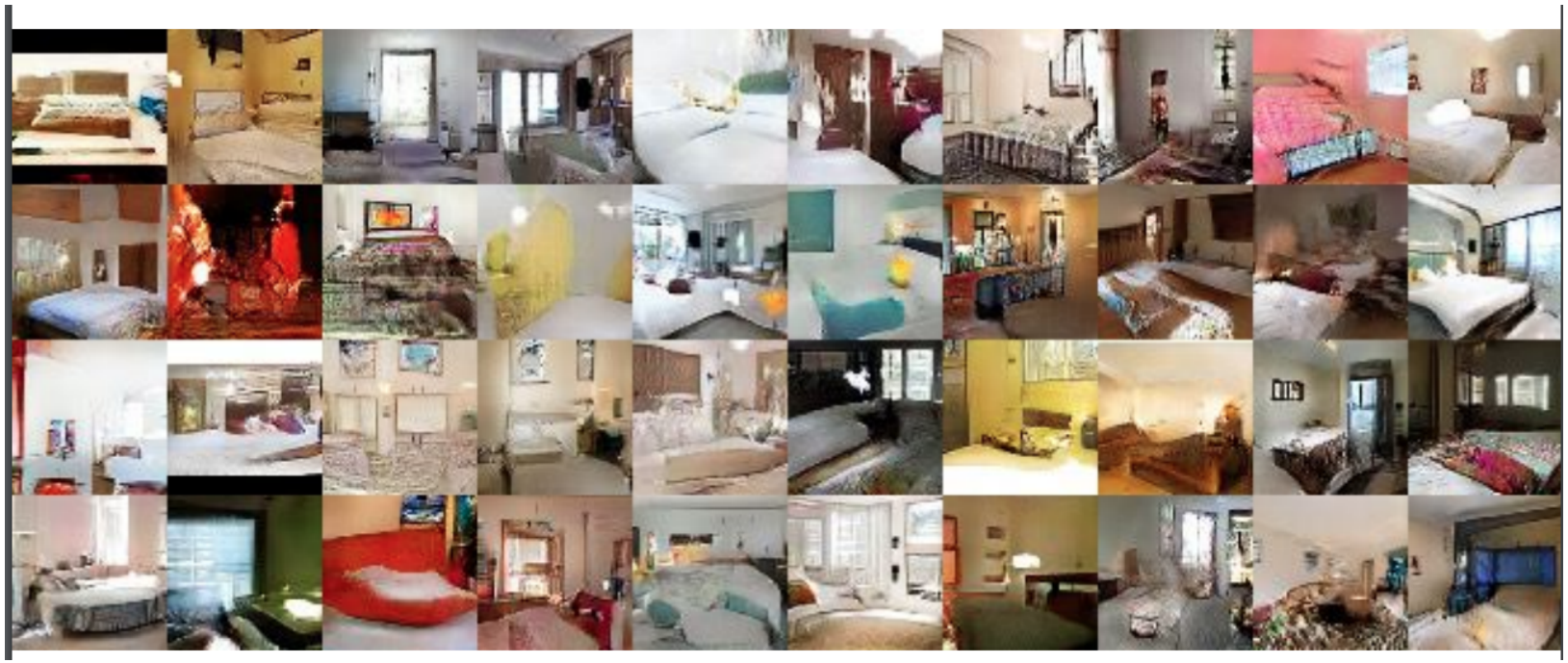
(Radford et al., 2015)

- **Idea:** Tricks to make GAN training more stable

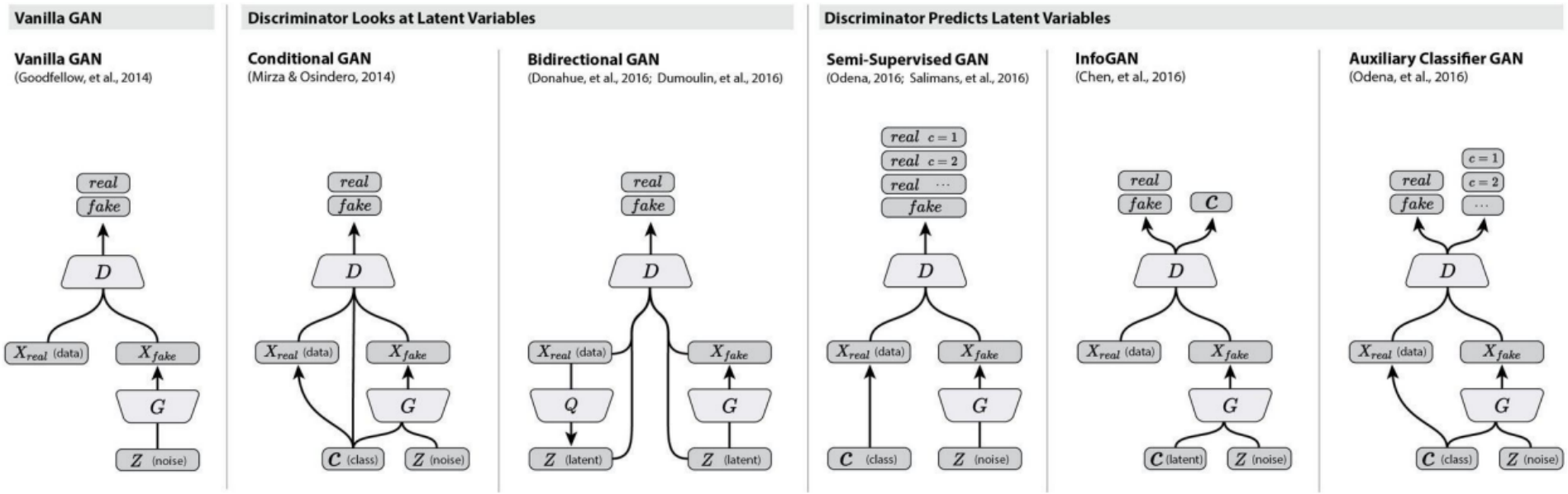


- No fully connected layers
- Batch Normalization (Ioffe and Szegedy, 2015)
- Leaky Rectifier in D
- Use Adam (Kingma and Ba, 2015)
- Tweak Adam hyperparameters a bit (lr=0.0002, b1=0.5)

Deep Convolutional GANs (DCGAN)

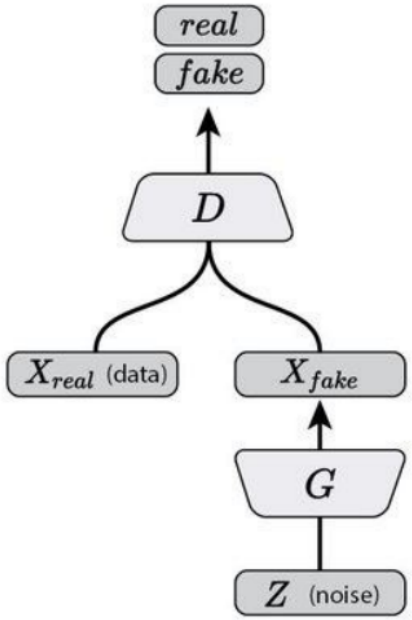


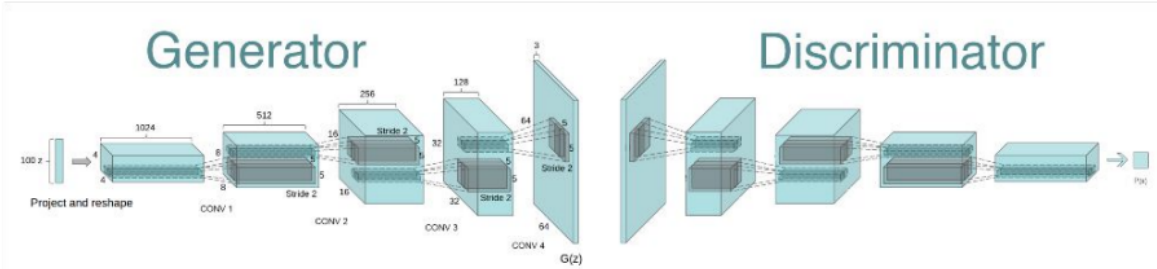
Subclasses of GANs



Vanilla GAN (Goodfellow et al., 2014

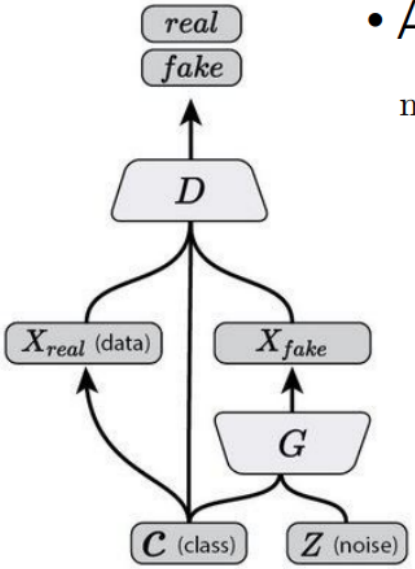
- Updating the generator:





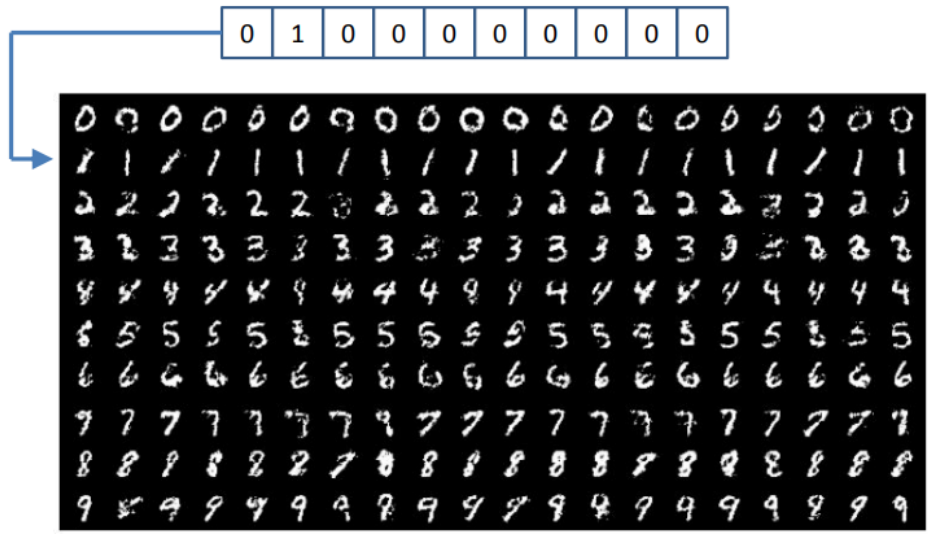
DCGAN (Radford et al., 2015)

Conditional GAN (Mirza et al)



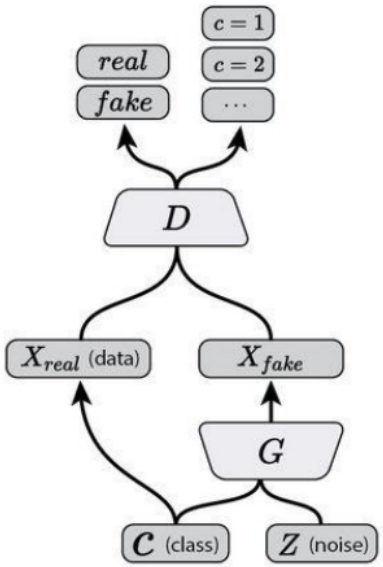
- Add conditional variables $\boldsymbol{y}$ into G and D

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$



3

Auxiliary Classifier GAN(Odena et al)



- Every generated sample has a corresponding class label

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]$$
$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})]$$

- D is trained to maximize $L_S + L_C$
- G is trained to maximize $L_C - L_S$

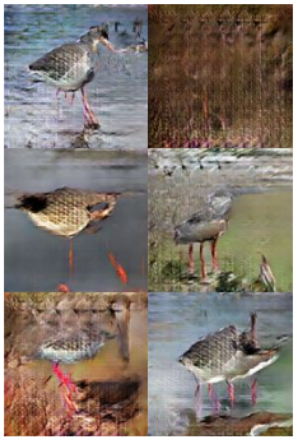
- Learns a representation for z that is independent of class label

Auxiliary Classifier GAN


monarch butterfly


goldfinch


daisy


redshank

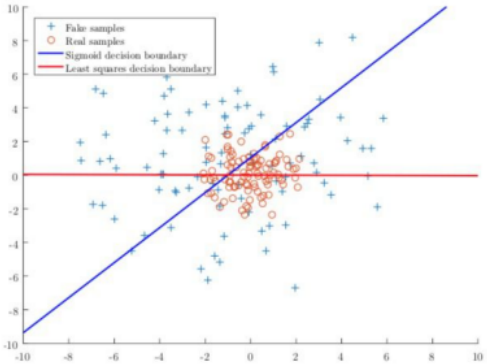

grey whale

Least Squares GAN (LSGAN)

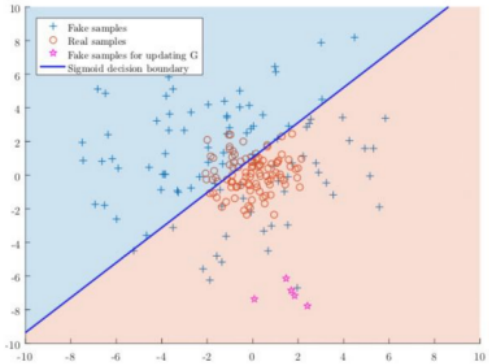
- Use a loss function that provides smooth and non-saturating gradient in discriminator D

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

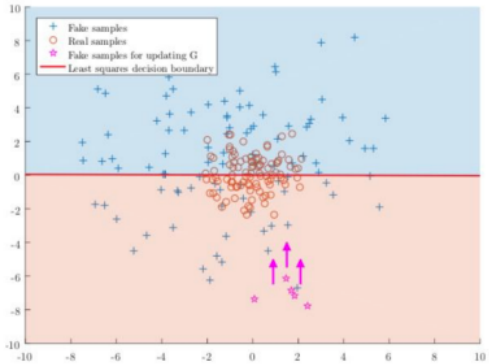
$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$



Decision boundaries of Sigmoid & Least Squares loss functions



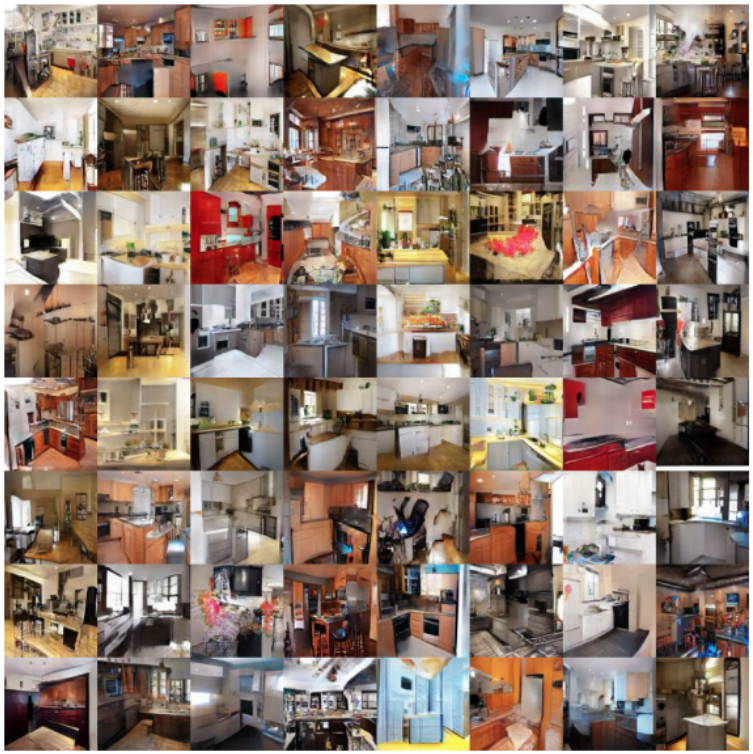
Sigmoid decision boundary



Least Squares decision boundary

Least Squares GAN (LSGAN)

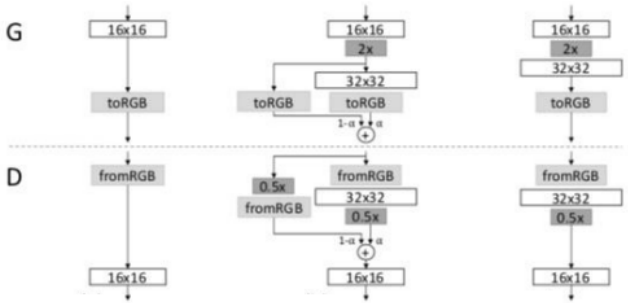
Church

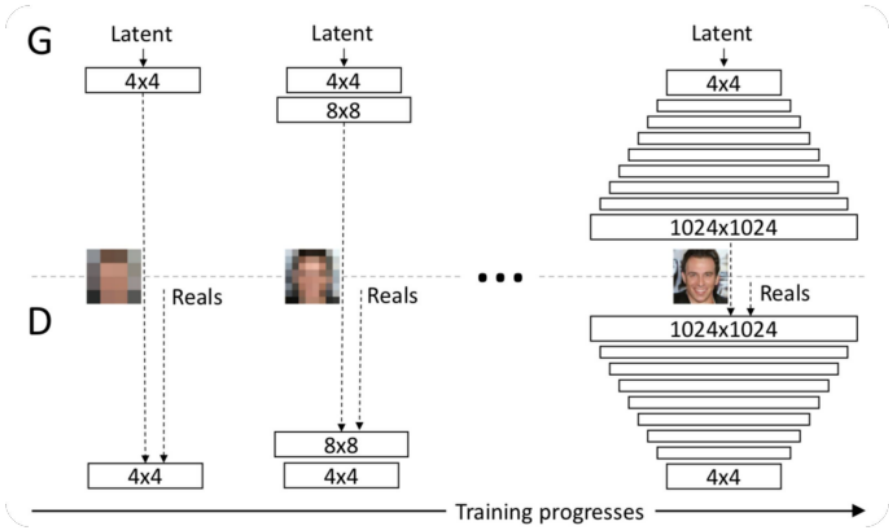


Kitchen

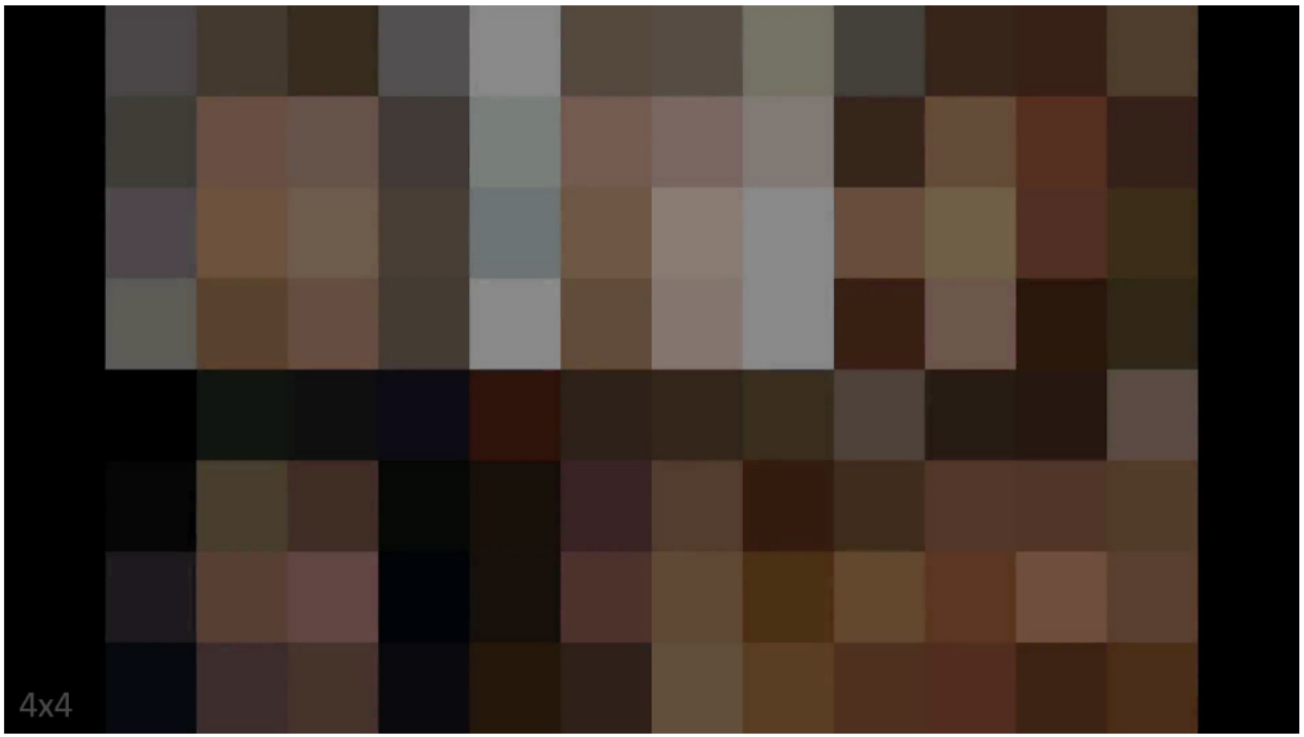
Progressive GANs(Karras et al., 2018)

- Progressively generate high-res images
- Multi-step training from low to high resolutions





Progressive GANs(Karras et al., 2018)

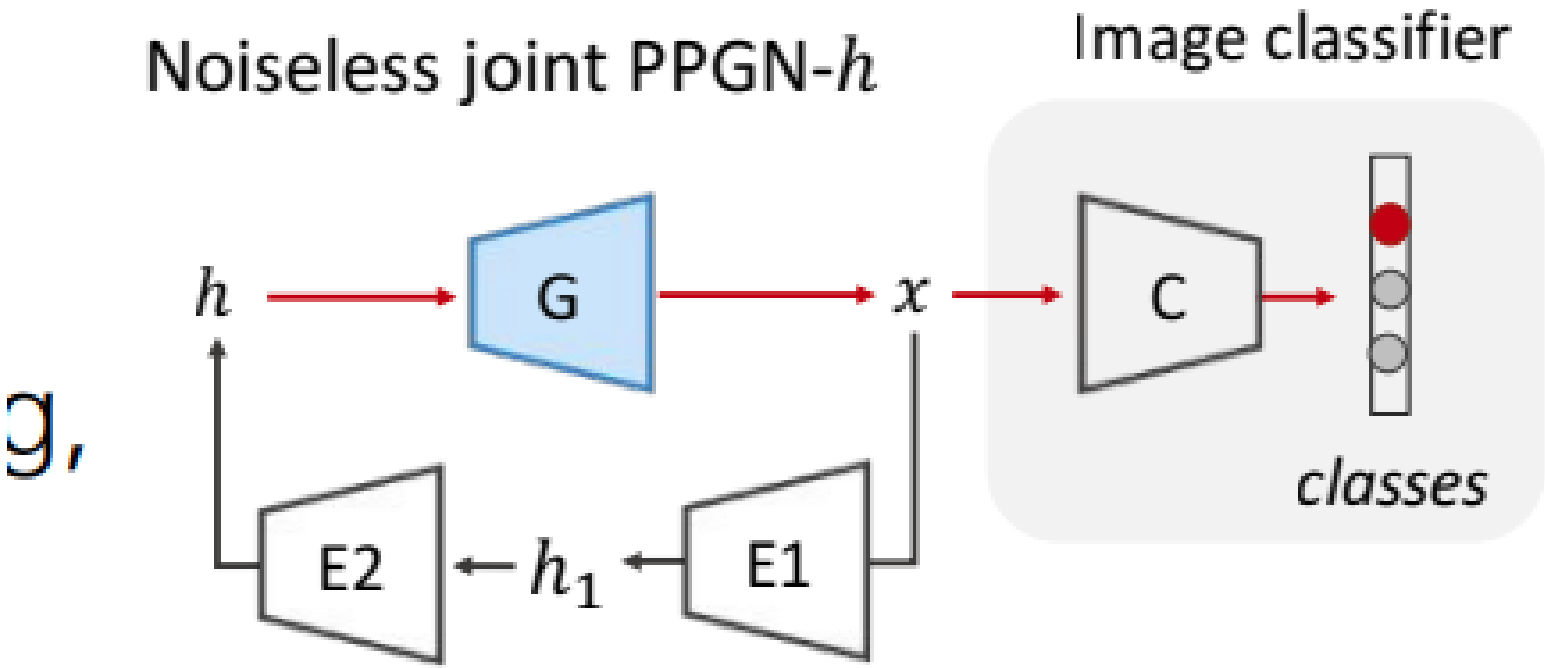


- Training process

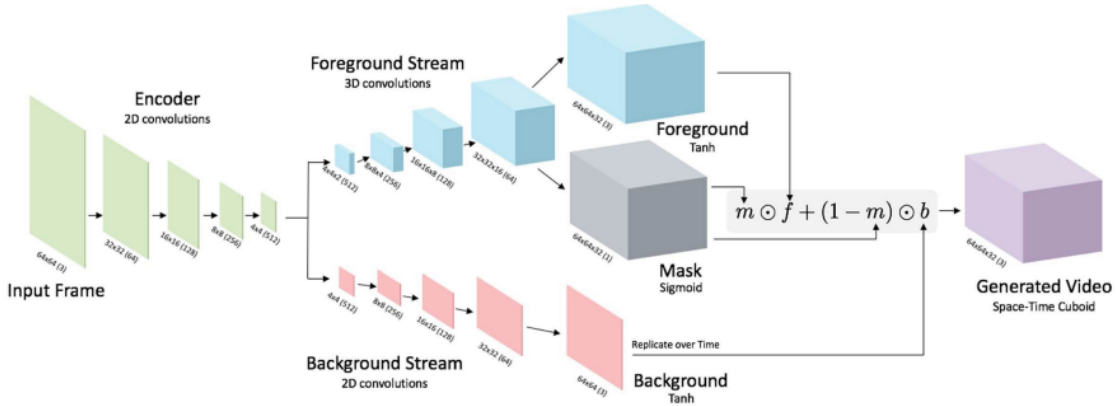
Progressive GANs(Karras et al., 2018)

Class-specific Image Generation (Nguyen et al., 2016)

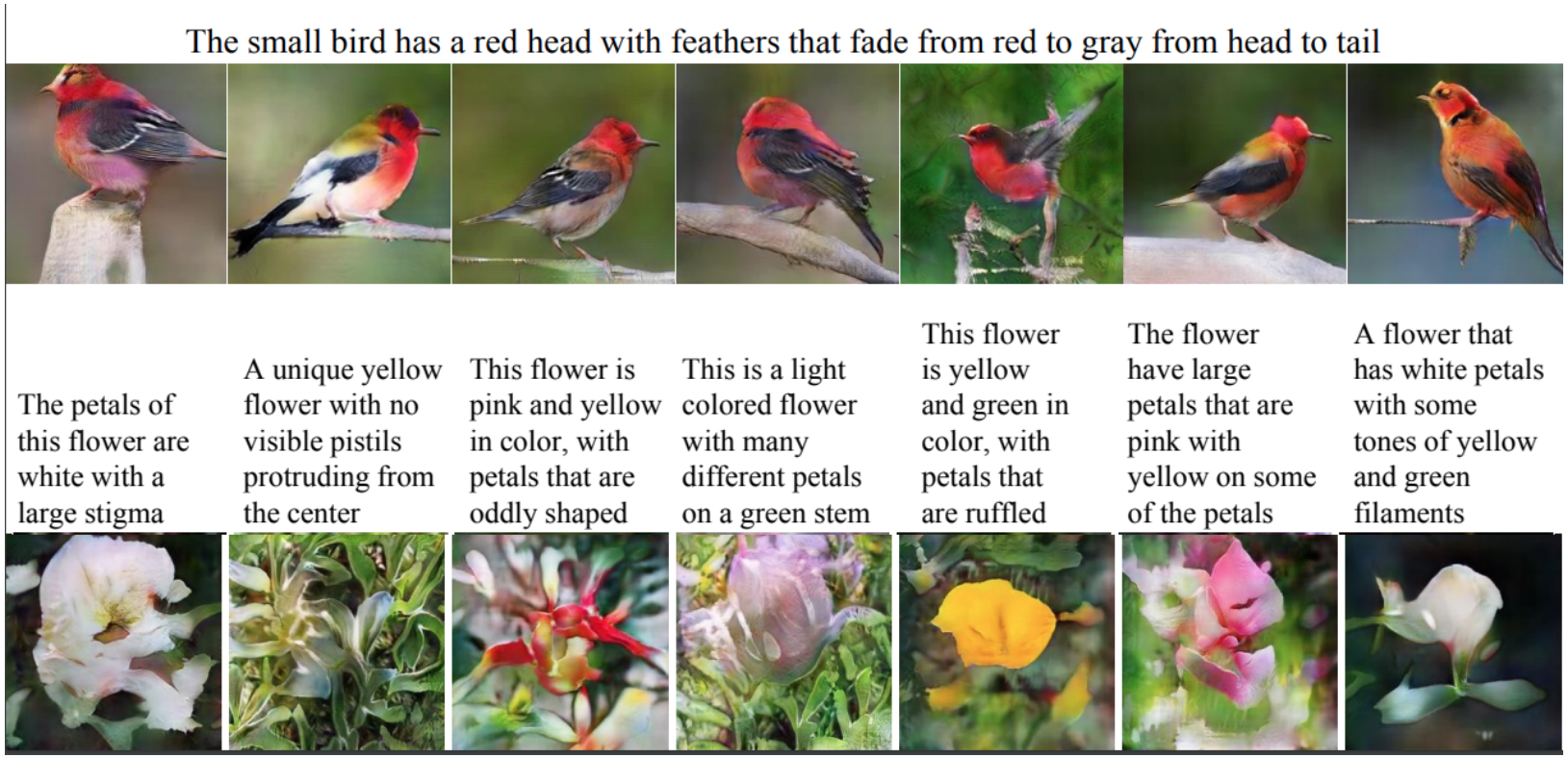
- Generates 227x227 realistic images from all ImageNet classes
- Combines adversarial training, moment matching, denoising autoencoders, and Langevin sampling



Video Generation (Vondrick et al., 2016)



Text-to-Image Synthesis (Zhang et al., 2016)



Text-to-Image Synthesis (Zhu et al., 2019)

Single Image Super-Resolution ((Ledig et al., 2016))

- Combine content loss with adversarial loss

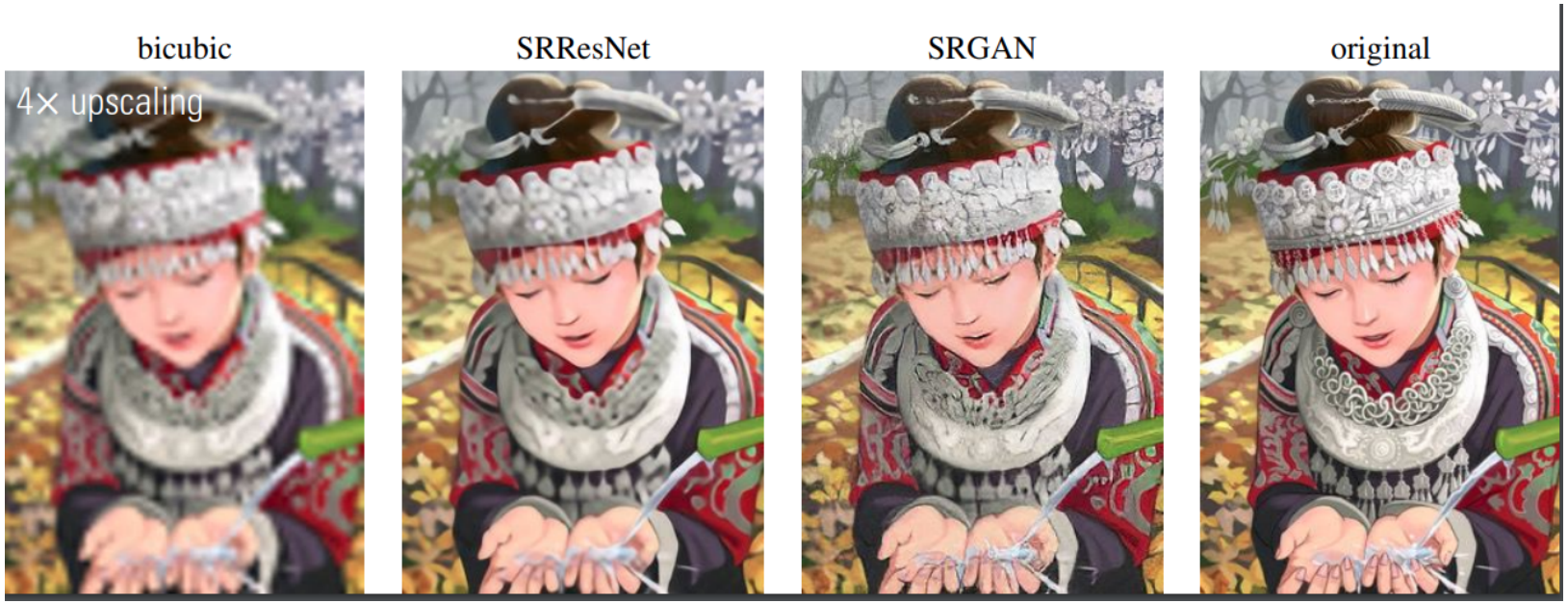


Image Inpainting (Pathak et al., 2016)

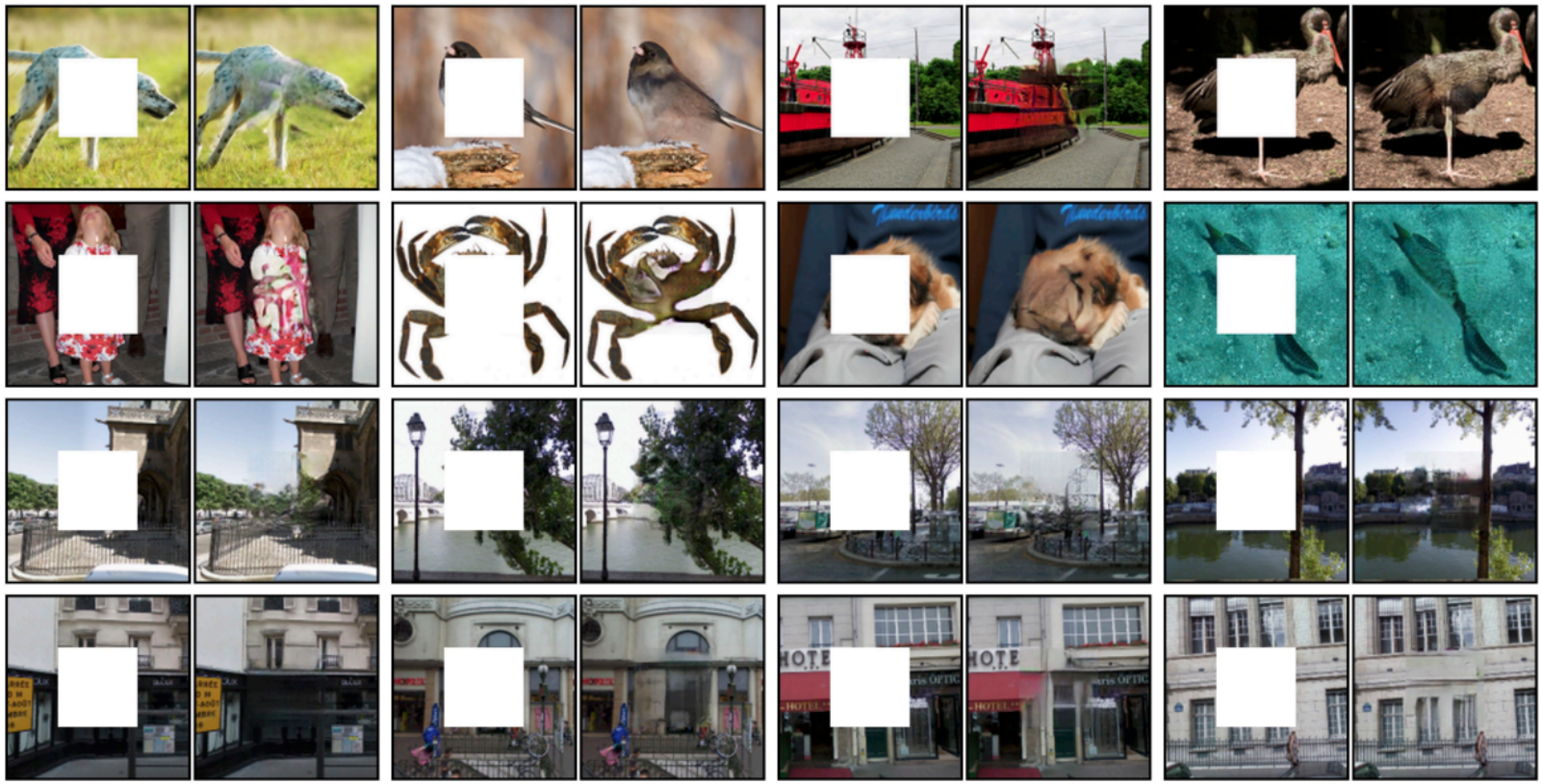


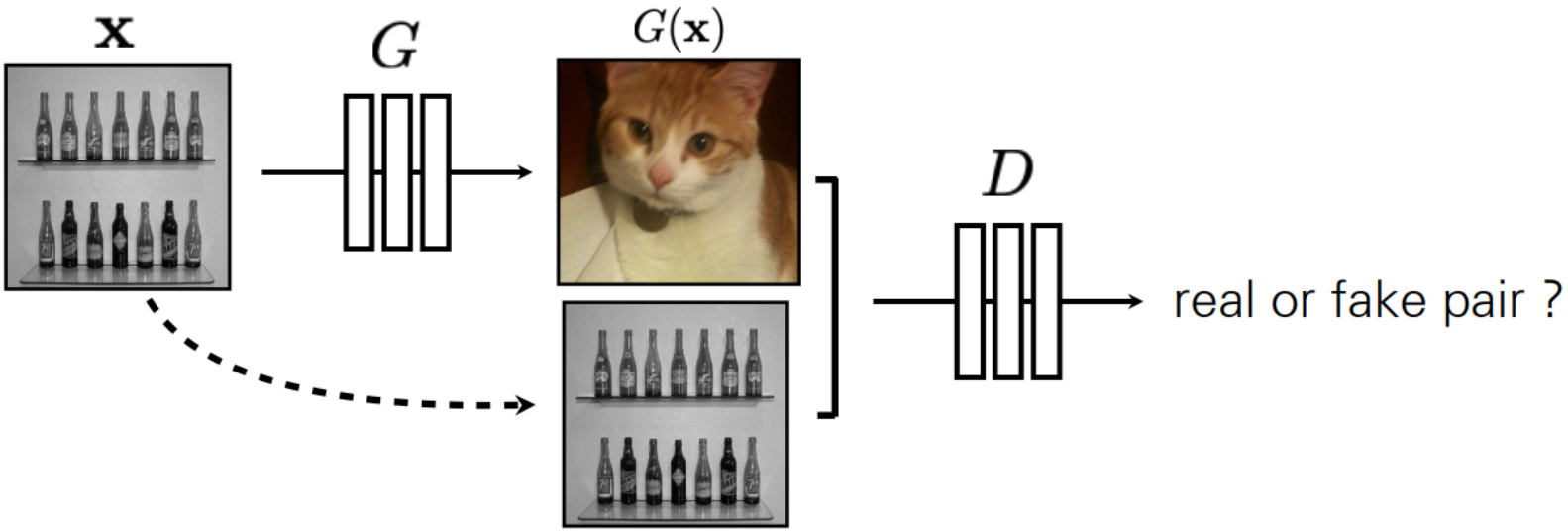
Image to Image Translation (Pix2Pix)

(Isola et al. 2016)

Image to Image Translation (Pix2Pix)

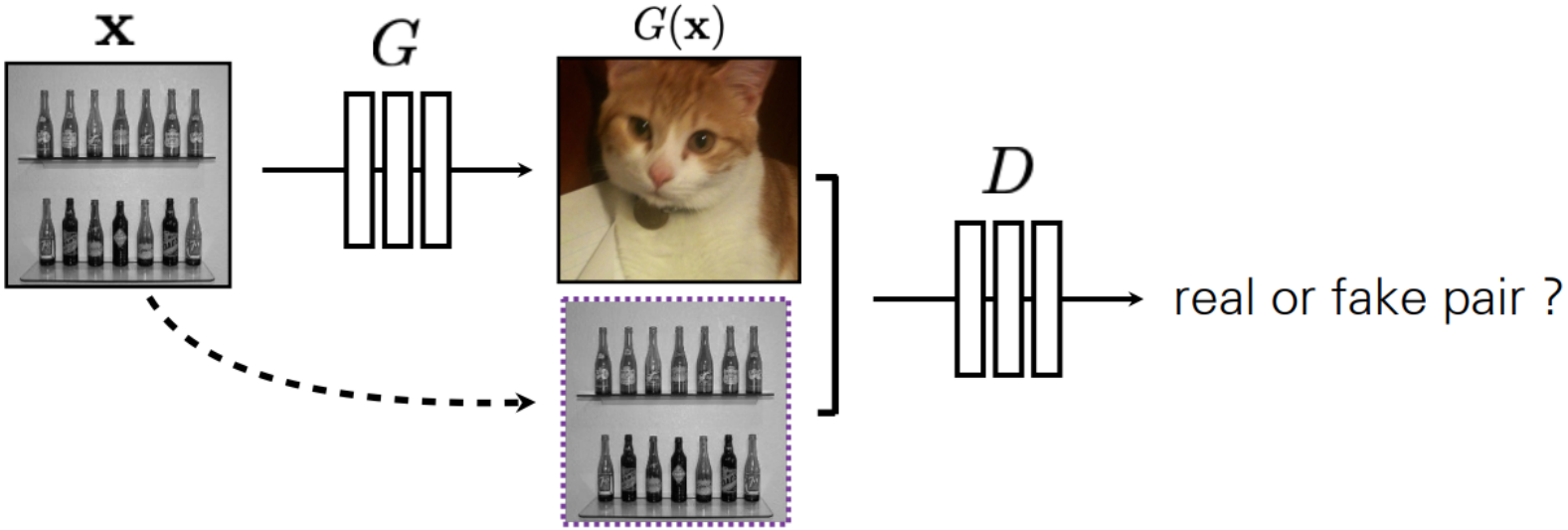
(Isola et al. 2016)

Image to Image Translation (Pix2Pix)



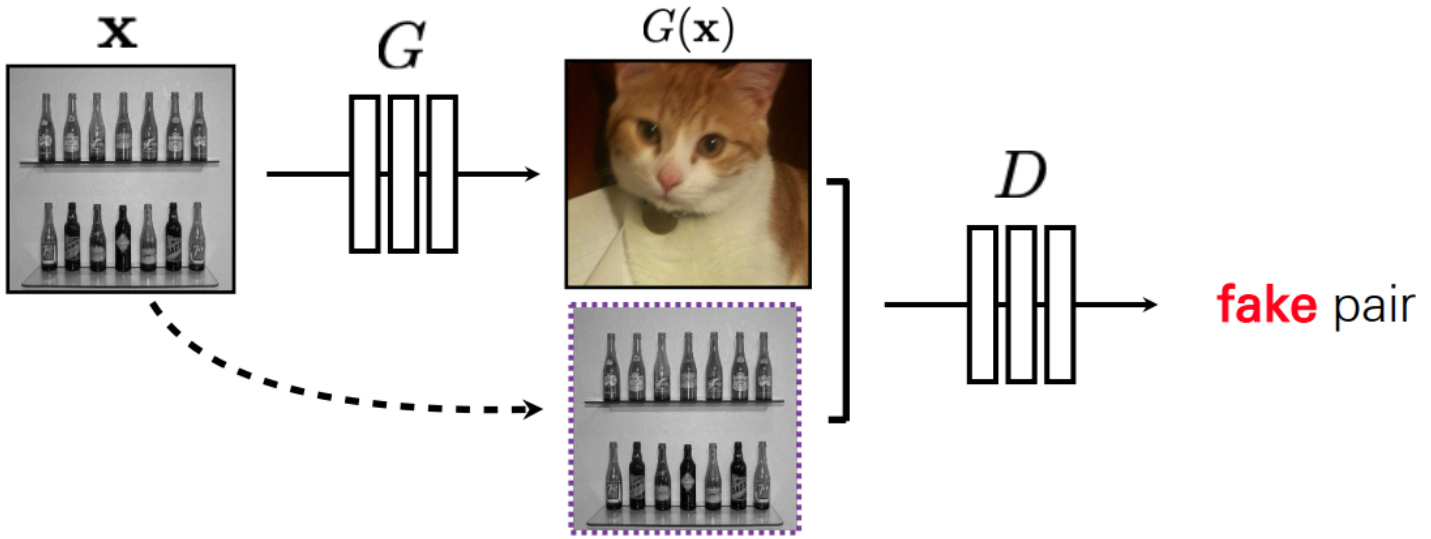
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \;+\; \log(1 - D(\mathbf{y}))\;]$$

Image to Image Translation (Pix2Pix)



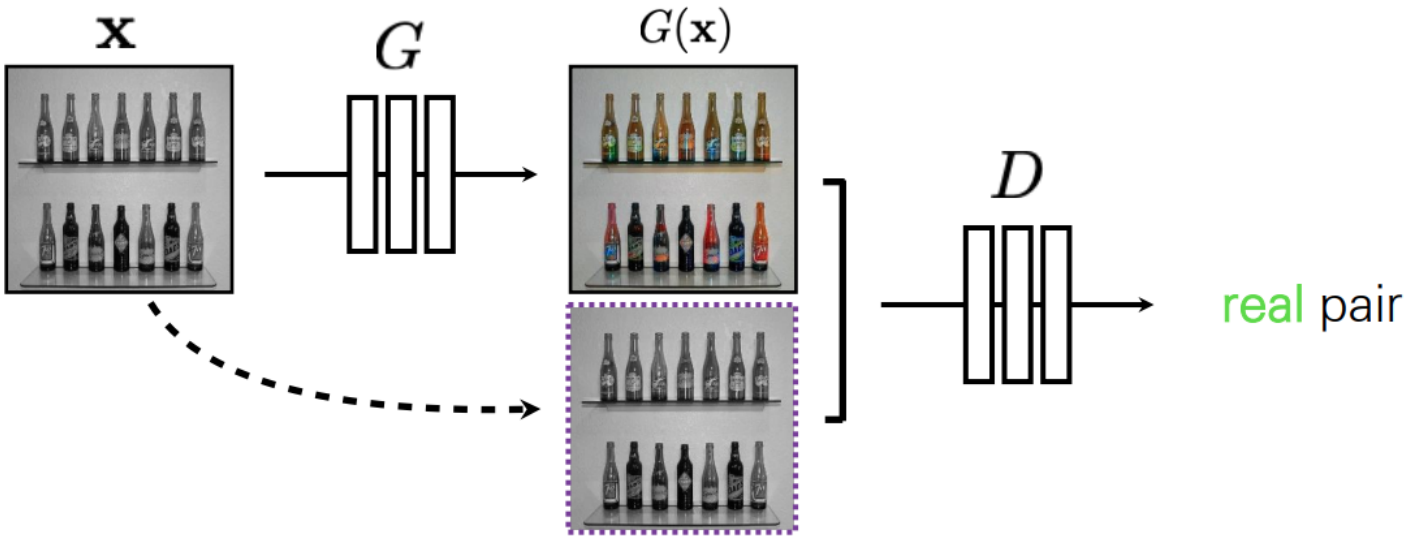
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

Image to Image Translation (Pix2Pix)



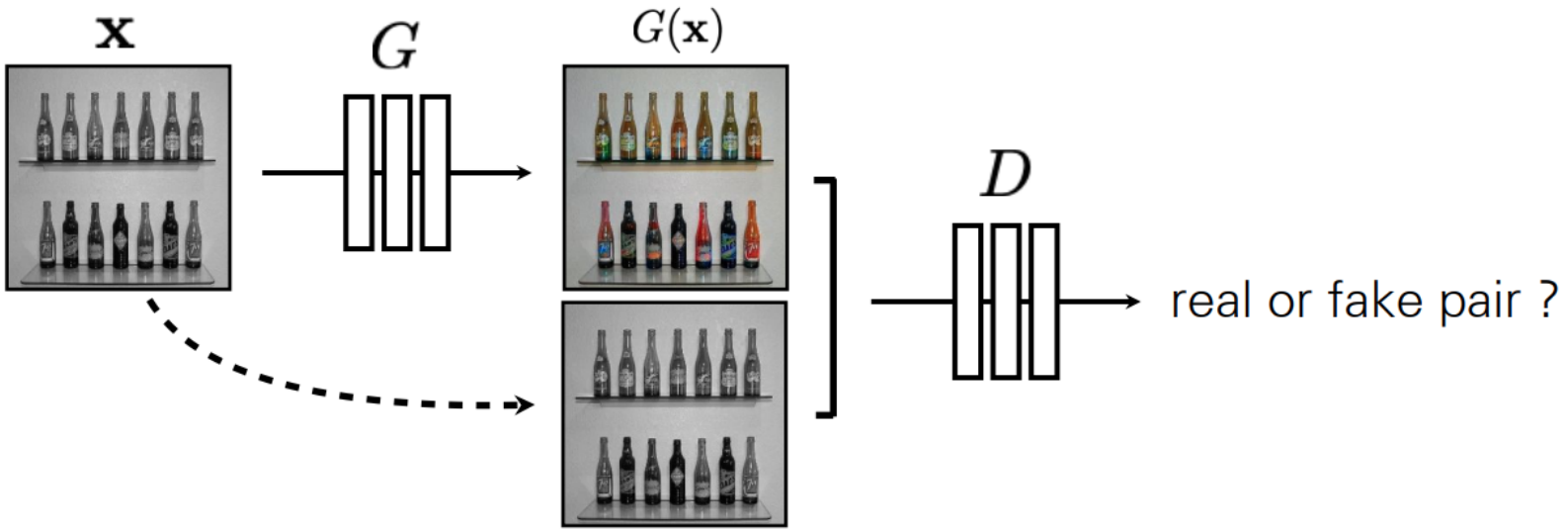
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

Image to Image Translation (Pix2Pix)



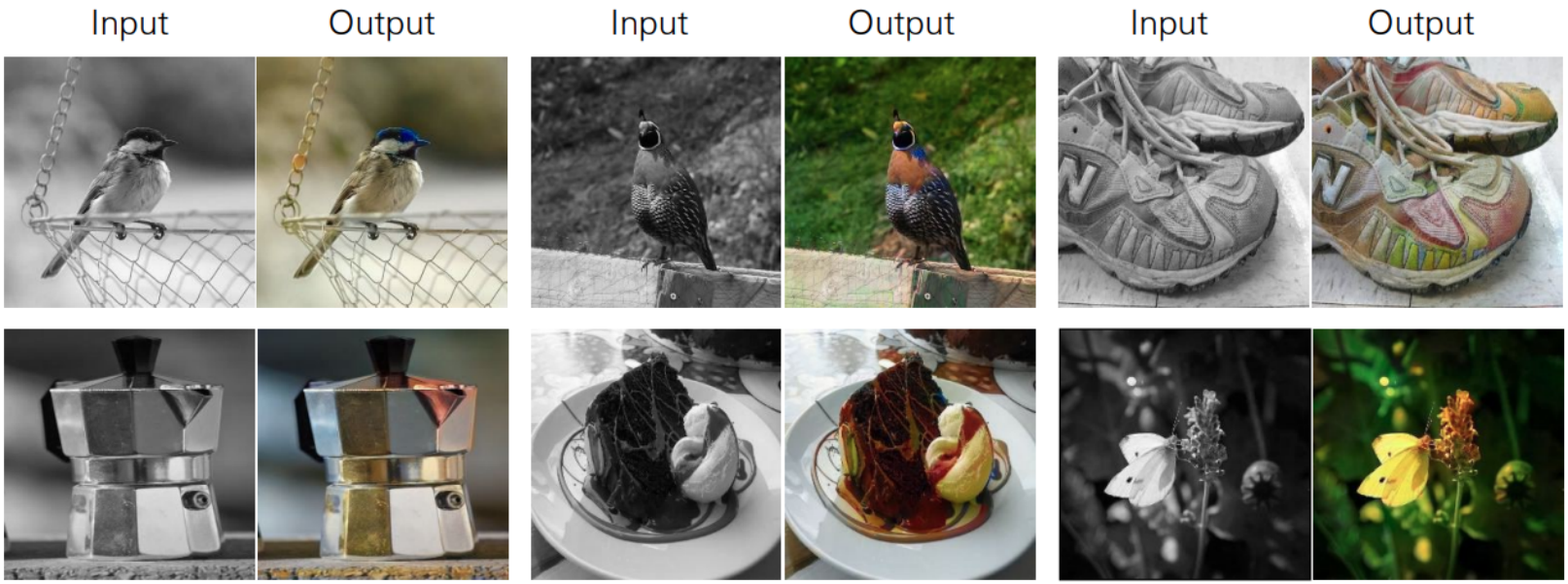
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

Image to Image Translation (Pix2Pix)

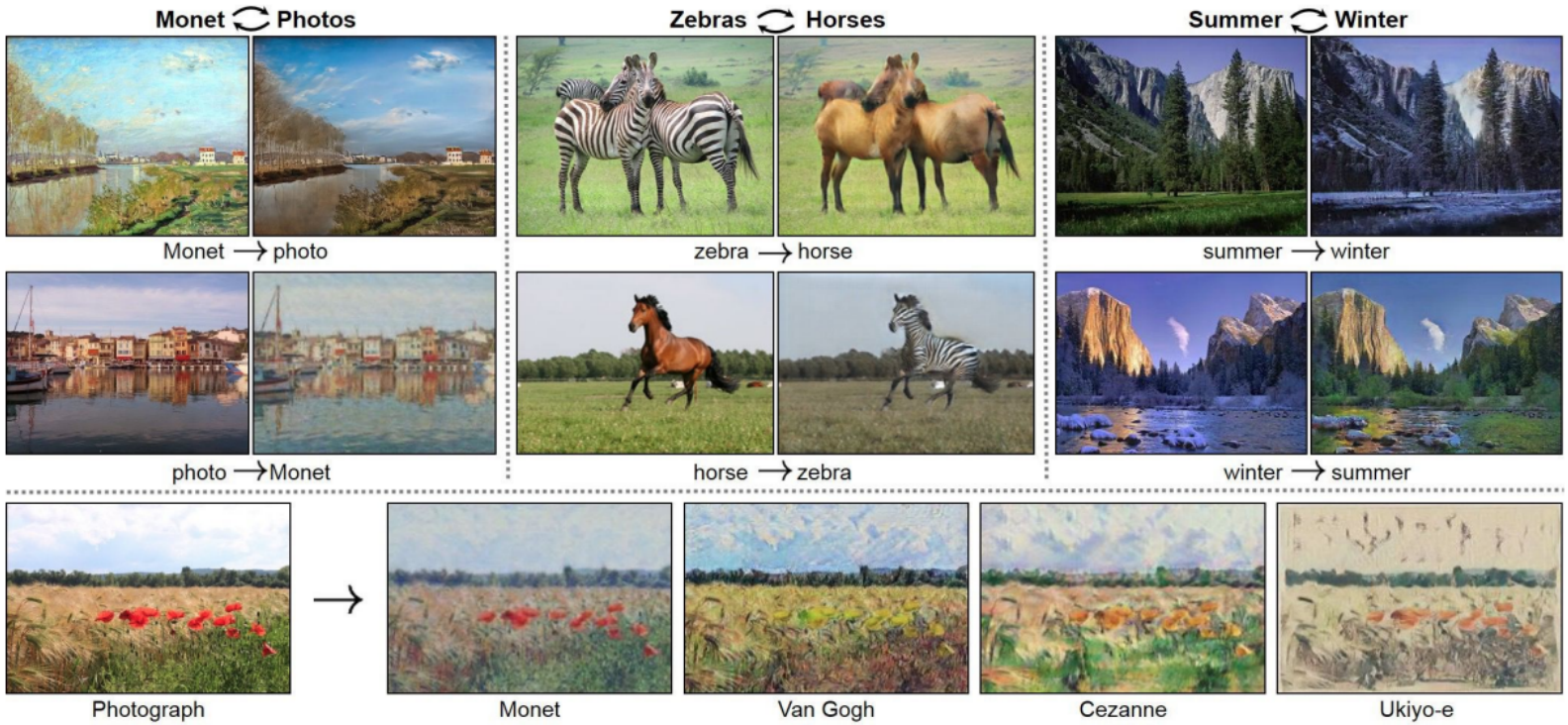


$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

BW –> Color

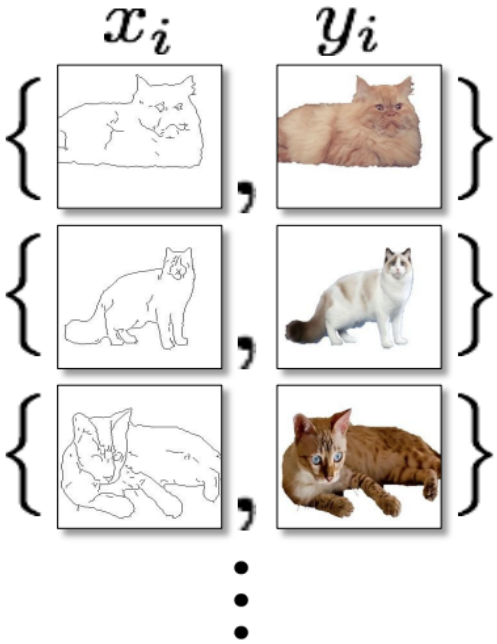


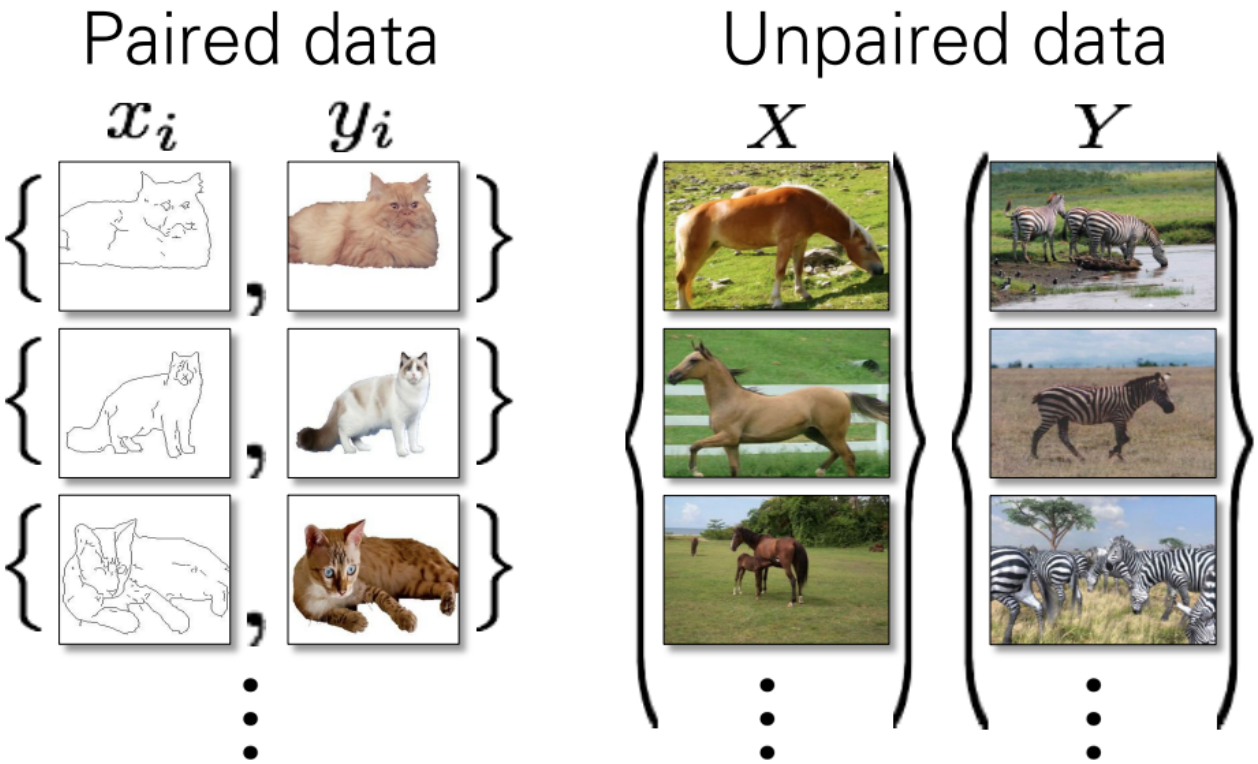
CycleGAN: Pix2Pix w/o input-output pairs

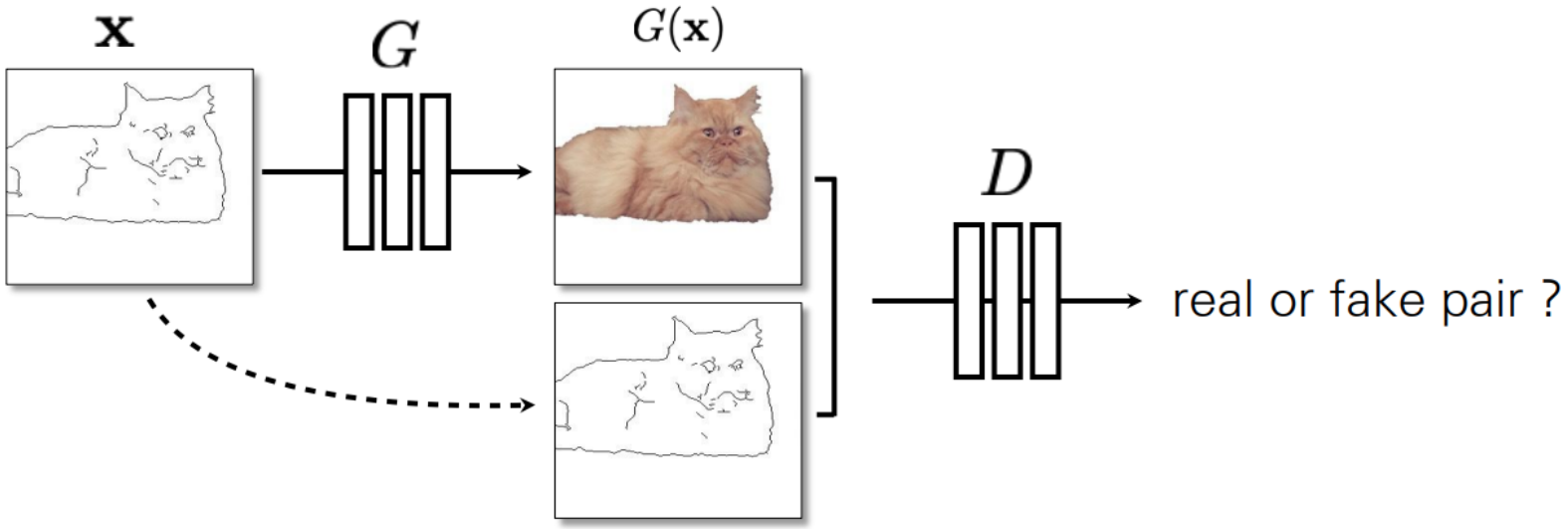


(Zhu et al. 2017)

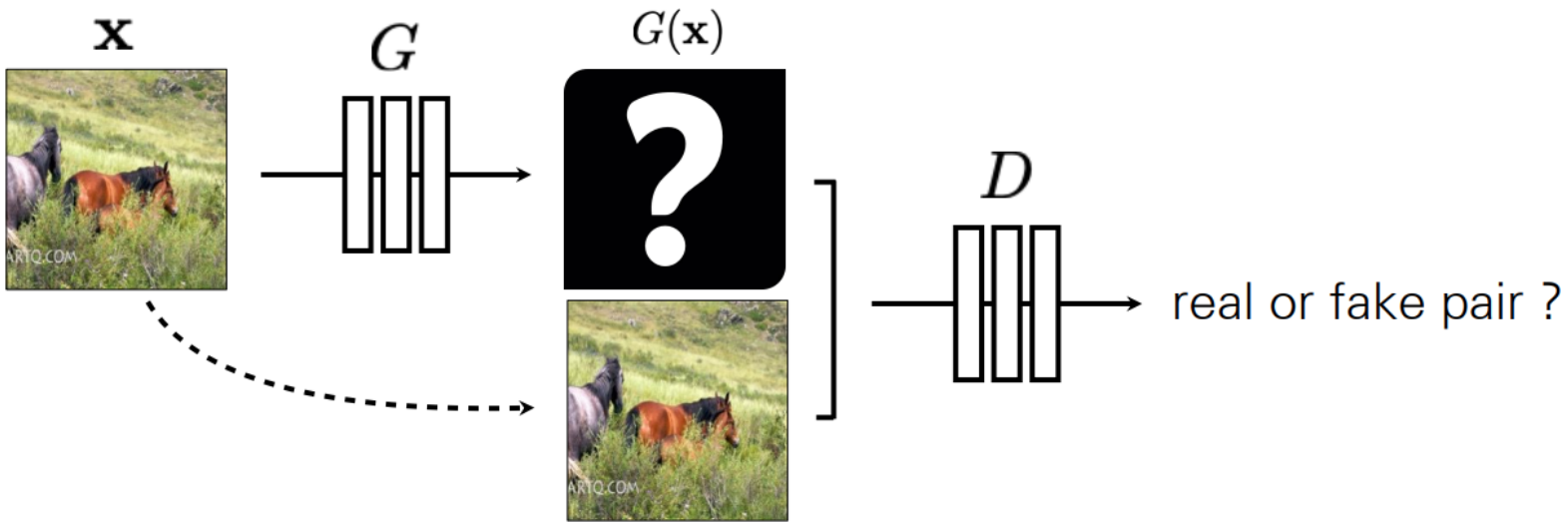
Paired data




Paired data
x_i
y_i
Unpaired data
X
Y

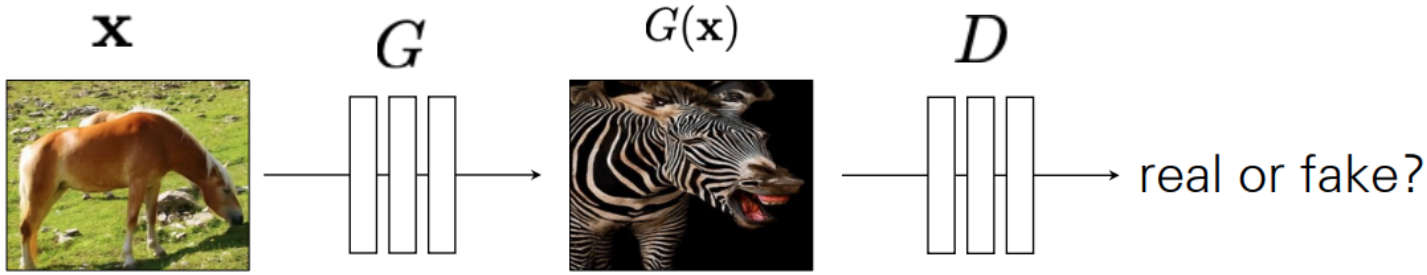


$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$



$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

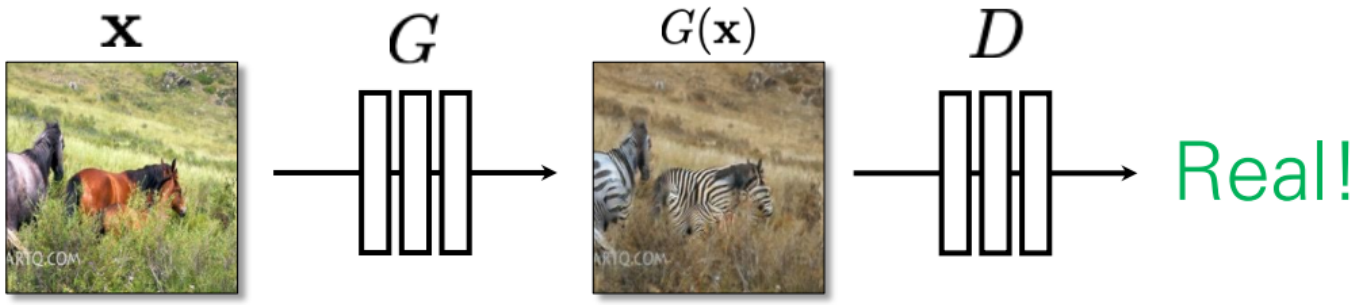
No input-output pairs!

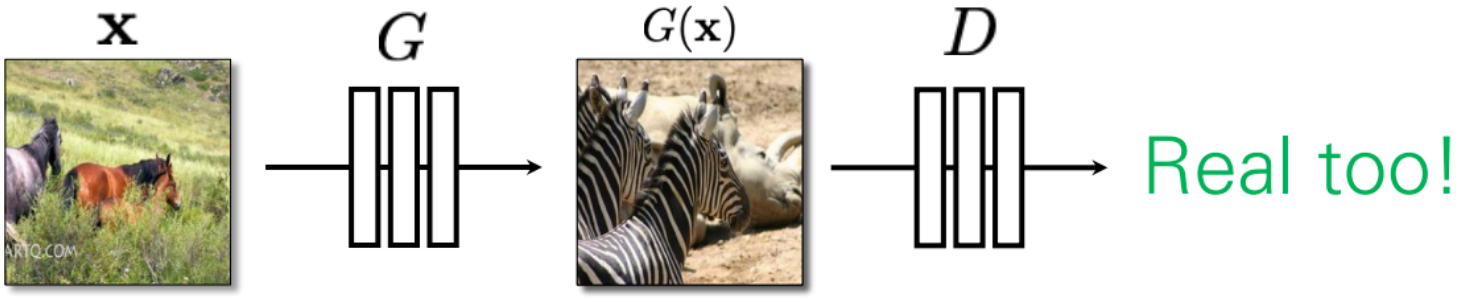


$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\;]$$

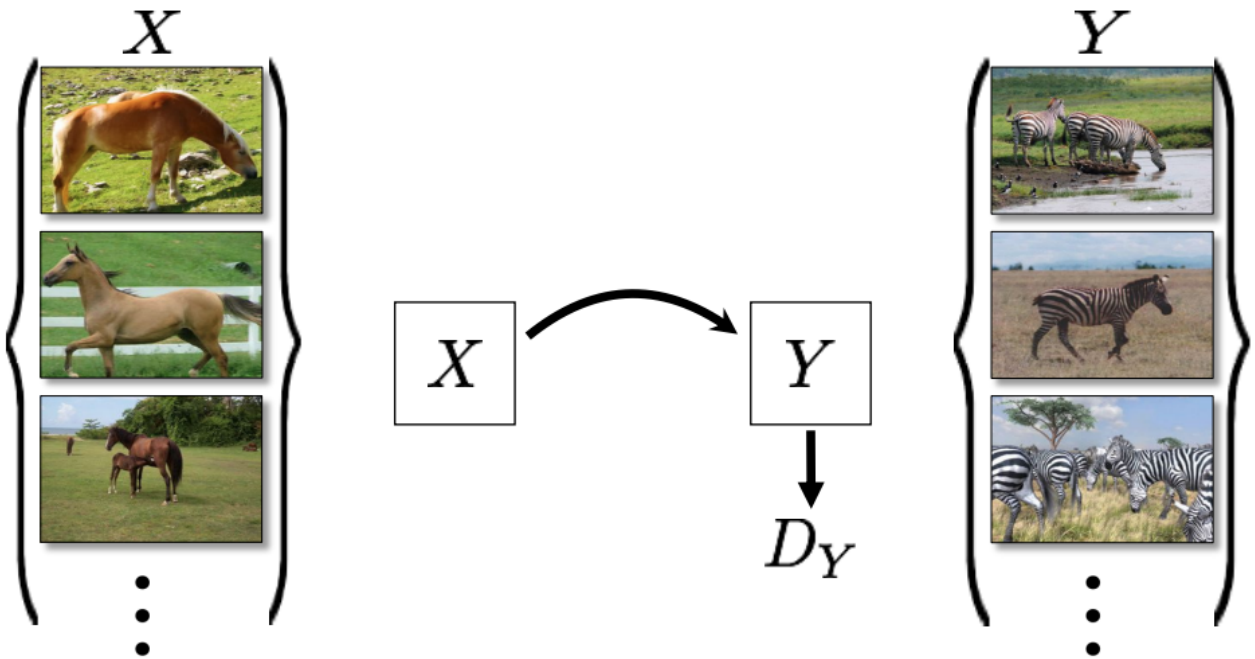
Usually loss functions check if output matches a target instance

GAN loss checks if output is part of an admissible set

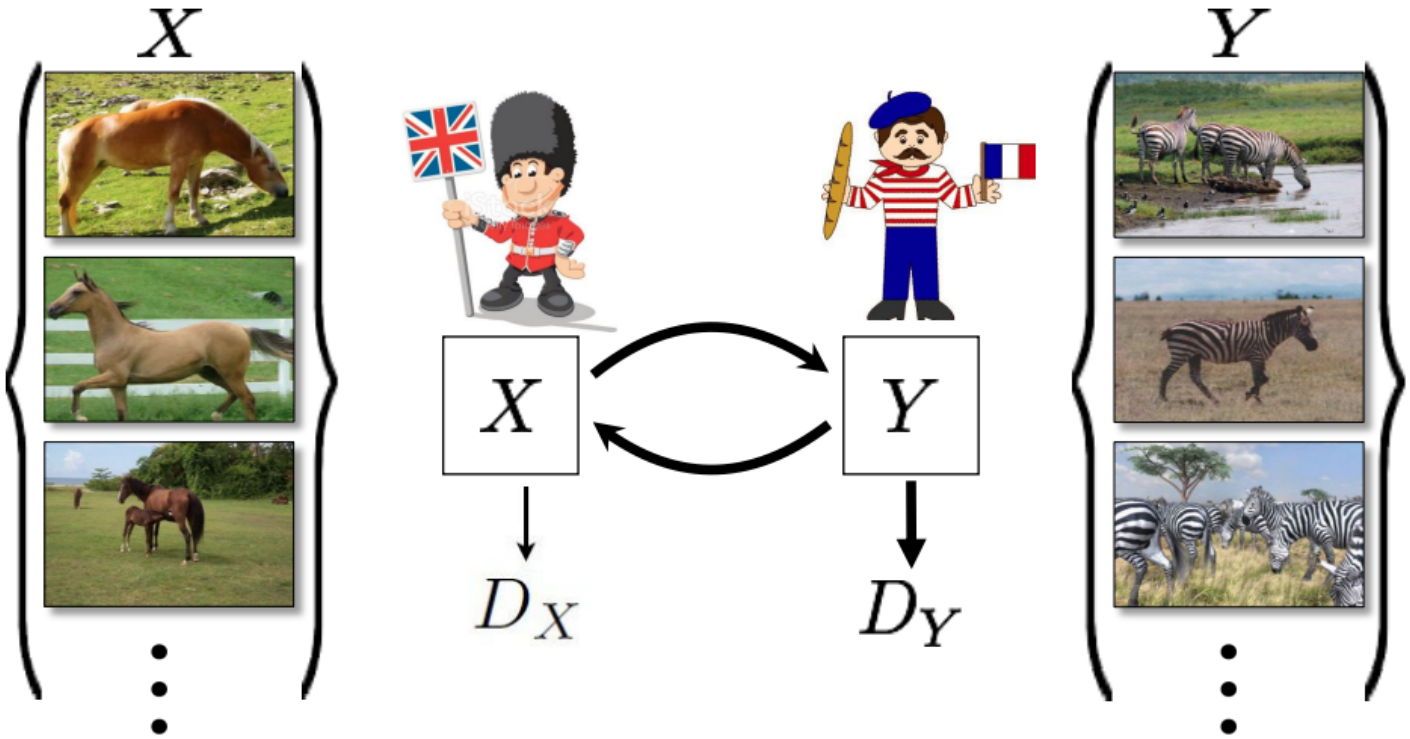

x
G
G(x)
D
Real!



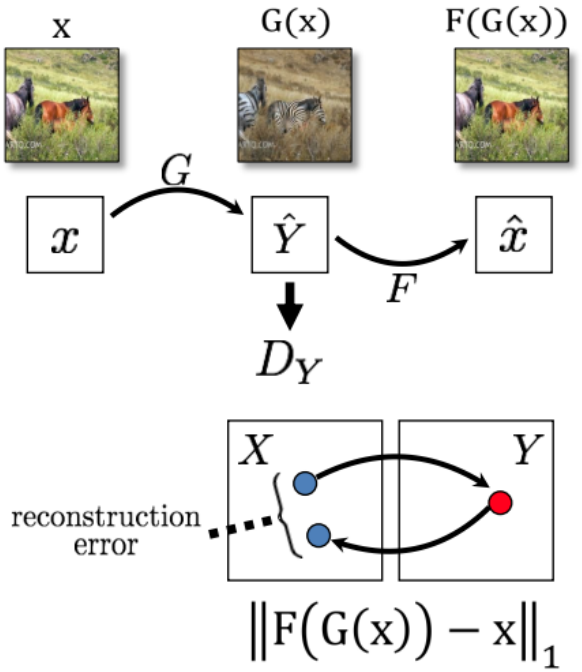
Nothing to force output to correspond to input



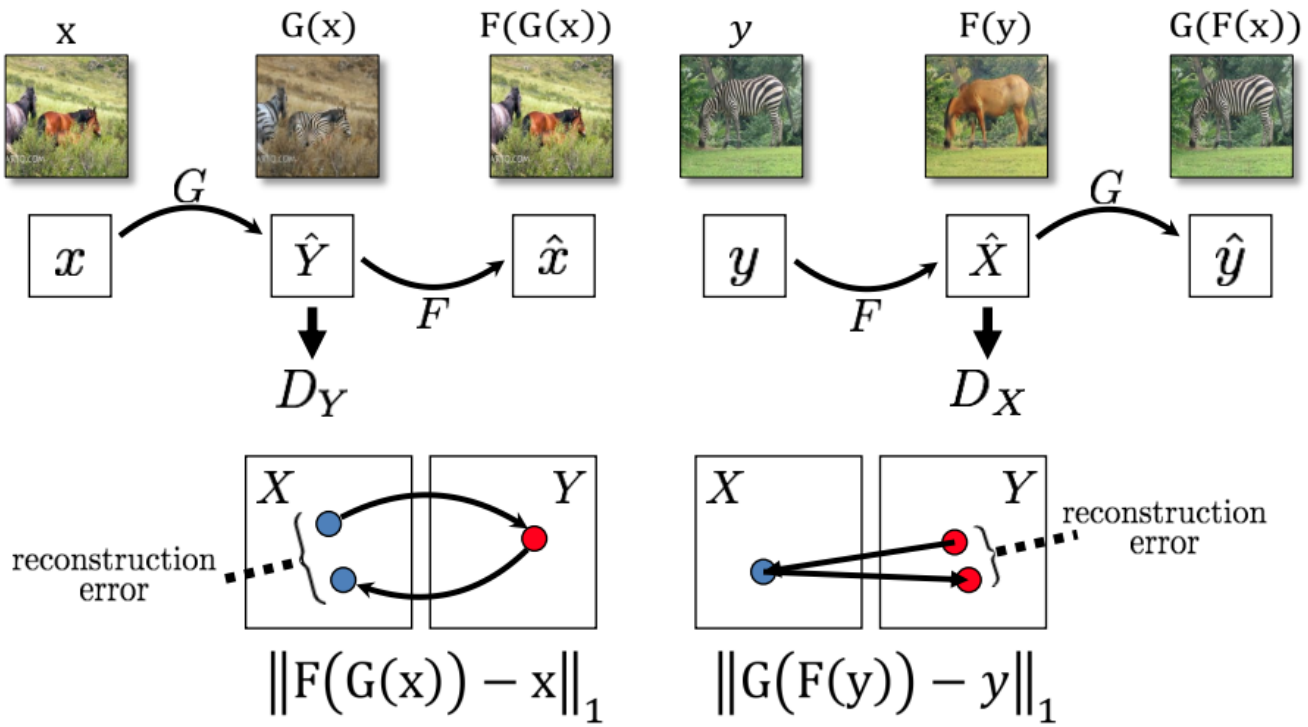
[Zhu et al. 2017], [Yi et al. 2017], [Kim et al. 2017]


X
X
Y
D_X
D_Y
Y

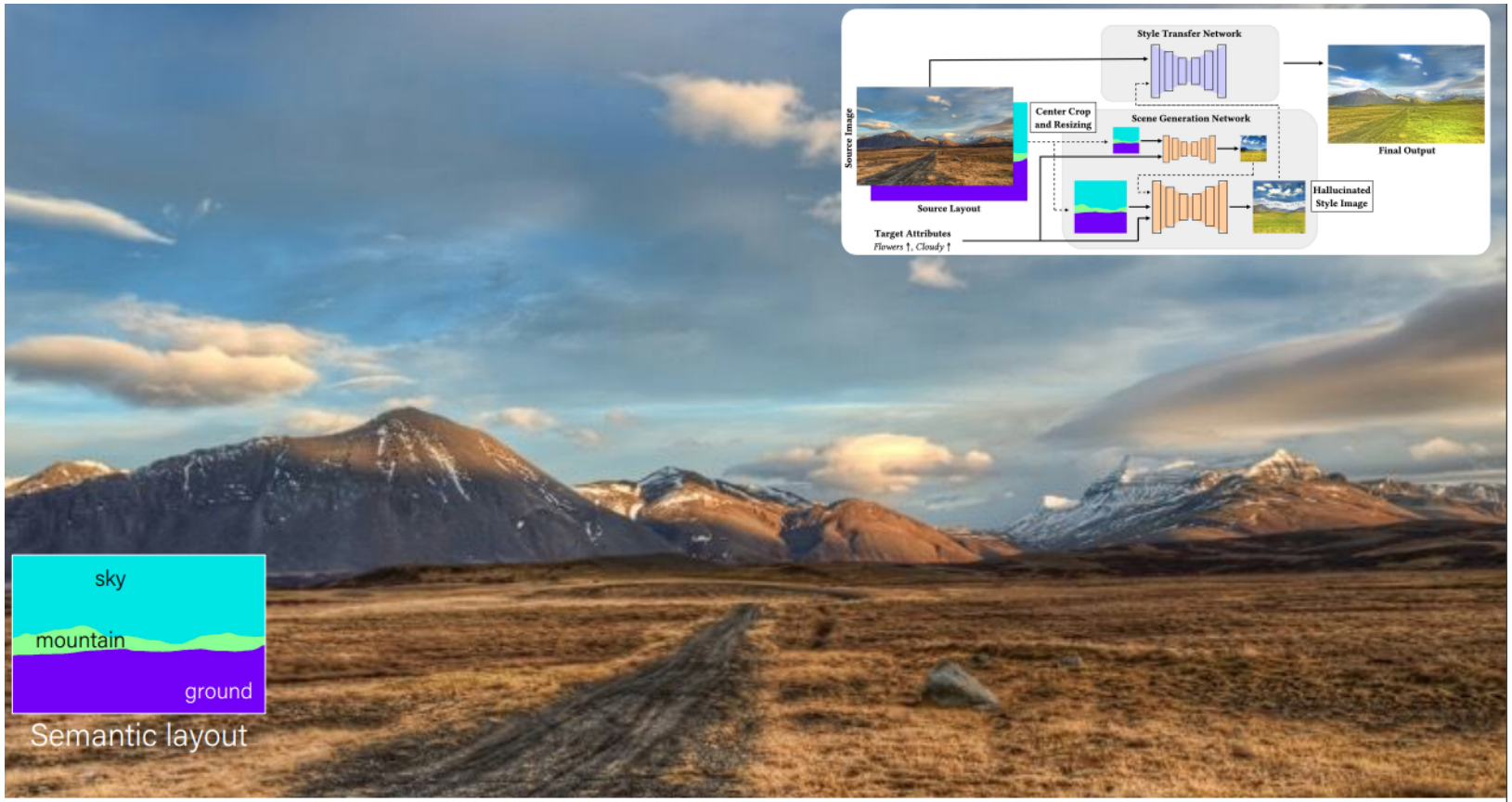
Cycle-Consistent Adversarial Networks



Cycle-Consistent Adversarial Networks



Cycle Consistency Loss

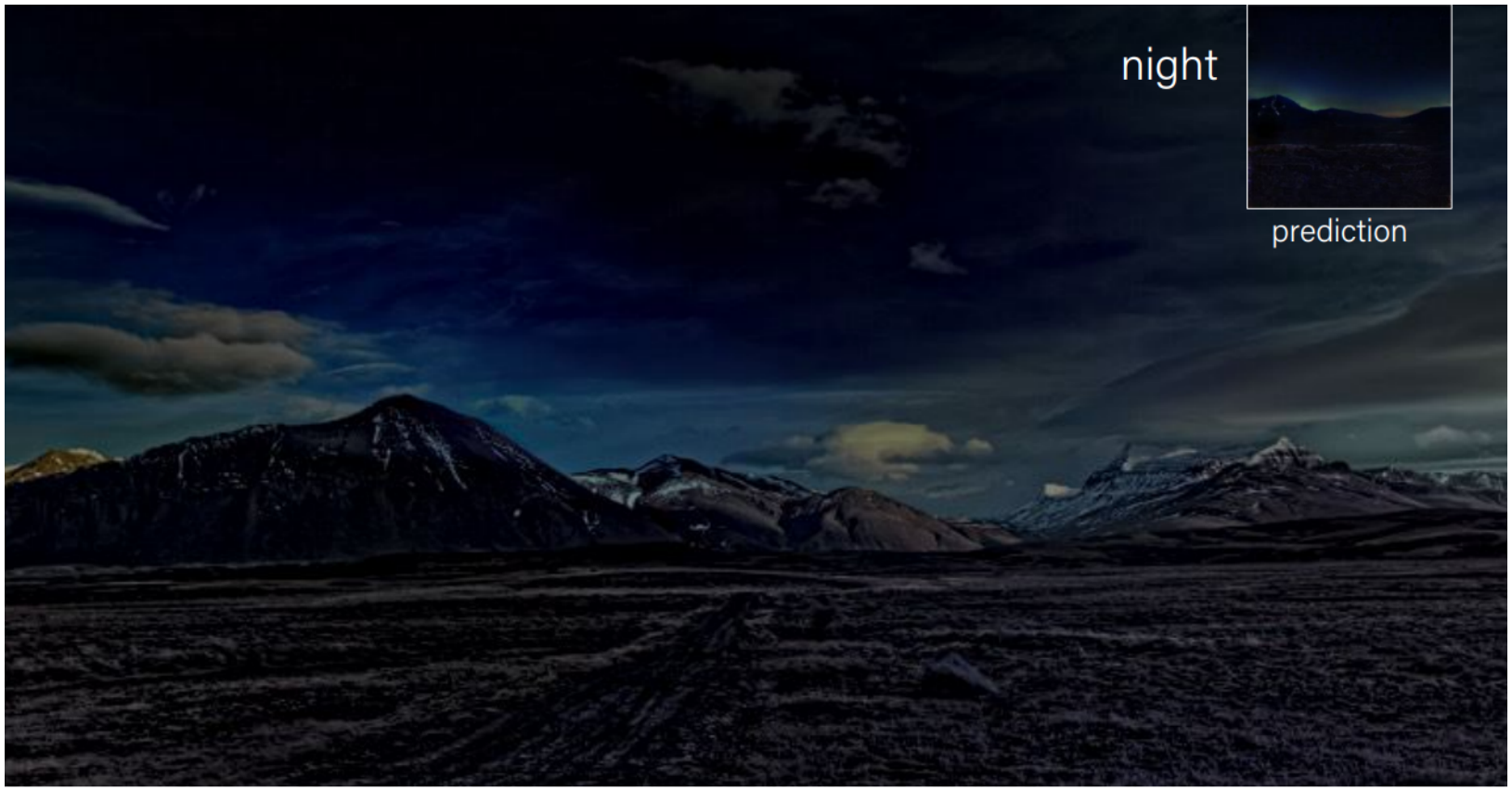


Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Cycle Consistency Loss

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Semantic Image Synthesis (SPADE)

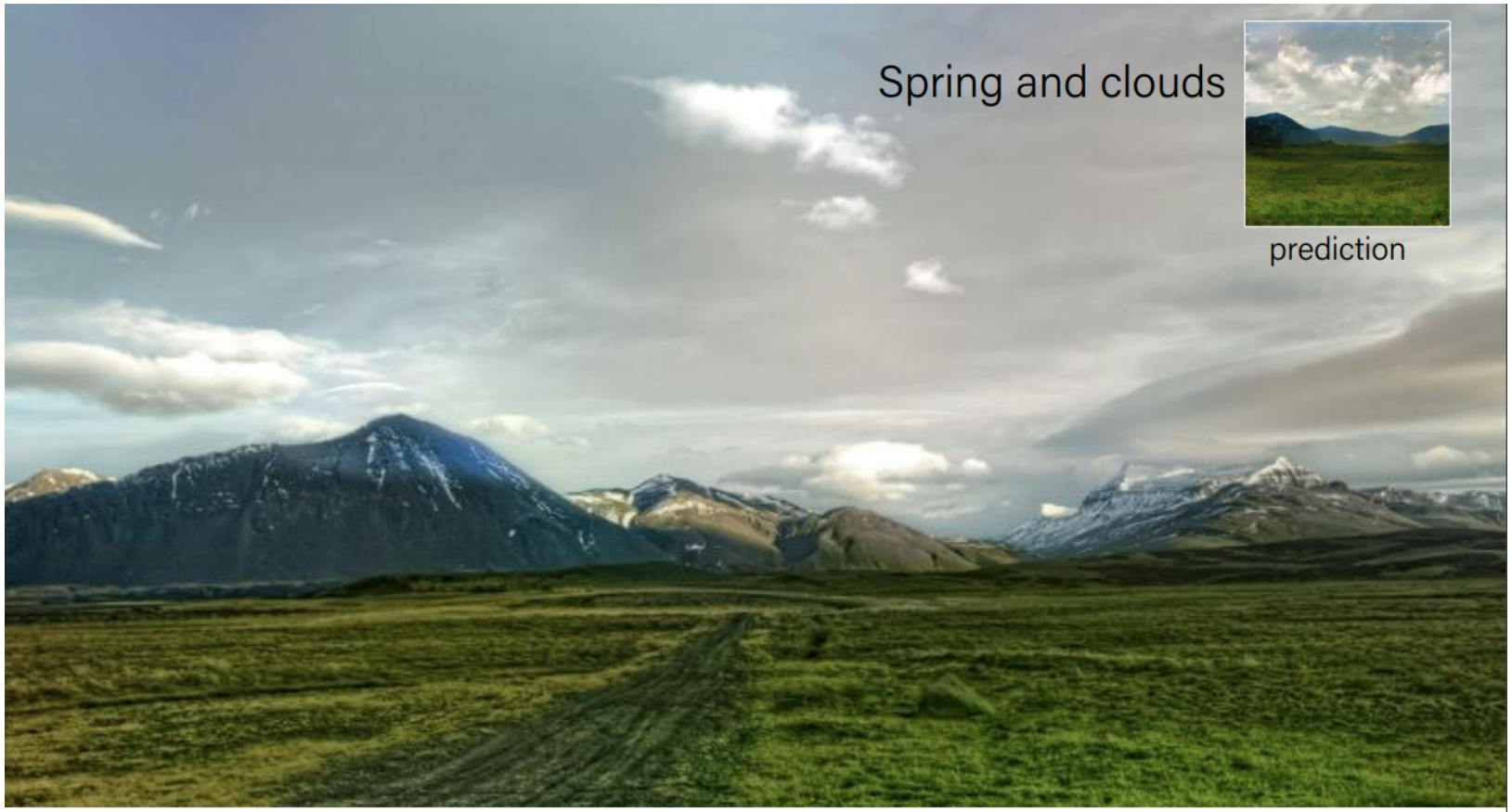


Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

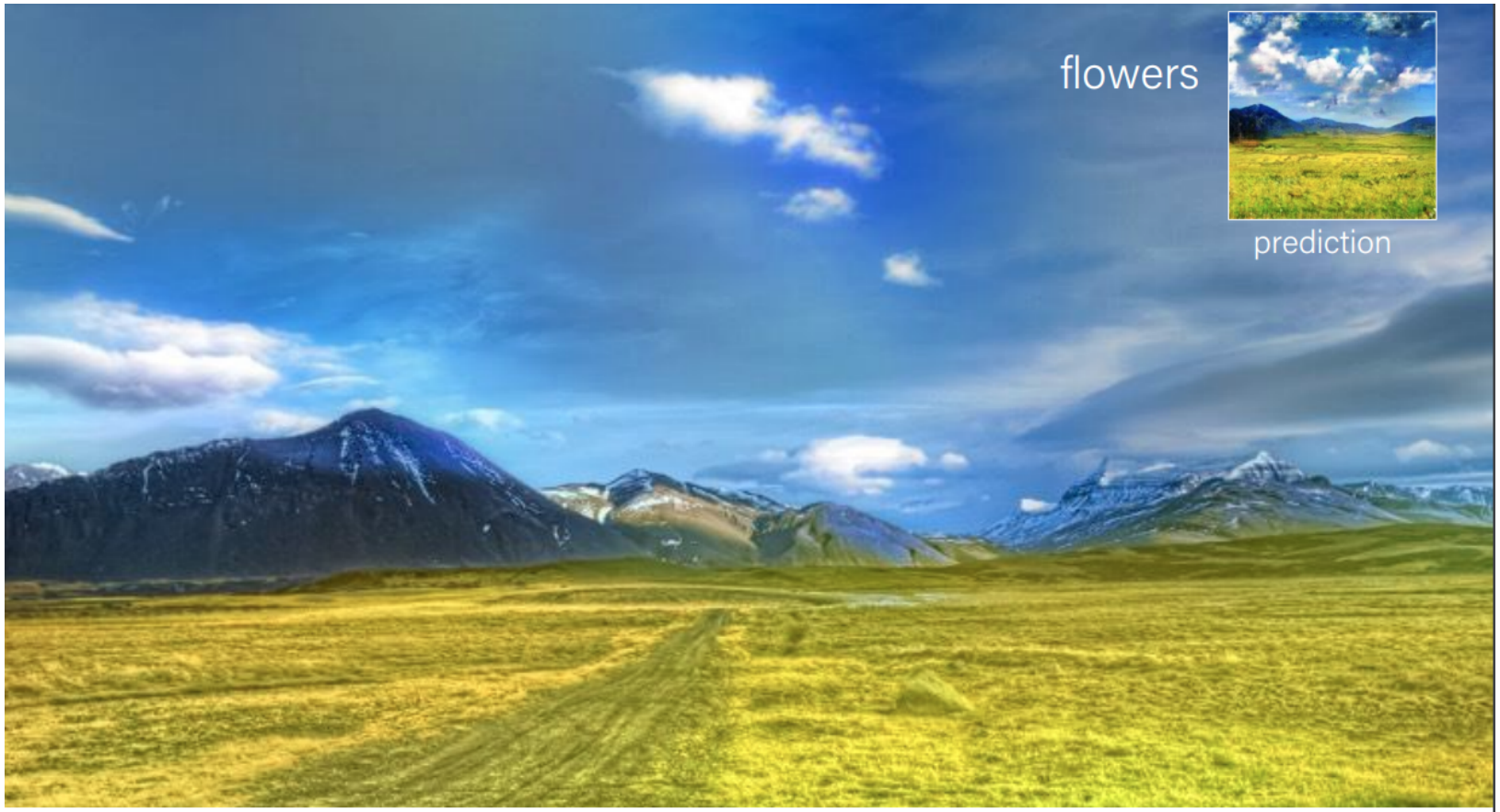
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]



Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]

Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2020]



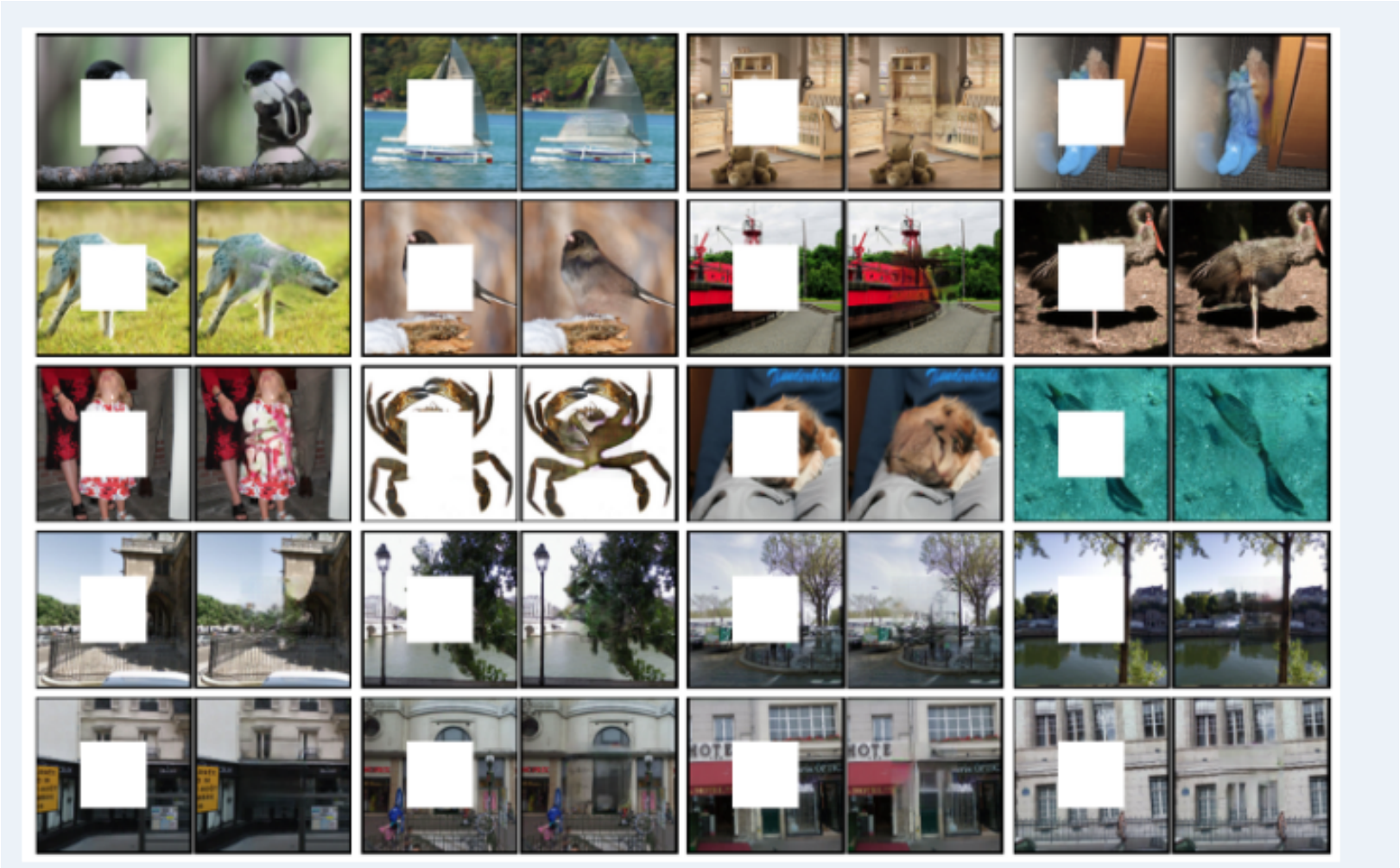
Manipulating Attributes of Natural Scenes via Hallucination [Karacan et al., 2018]

Context Encoder: Feature Learning by Inpainting (Pathak et al)

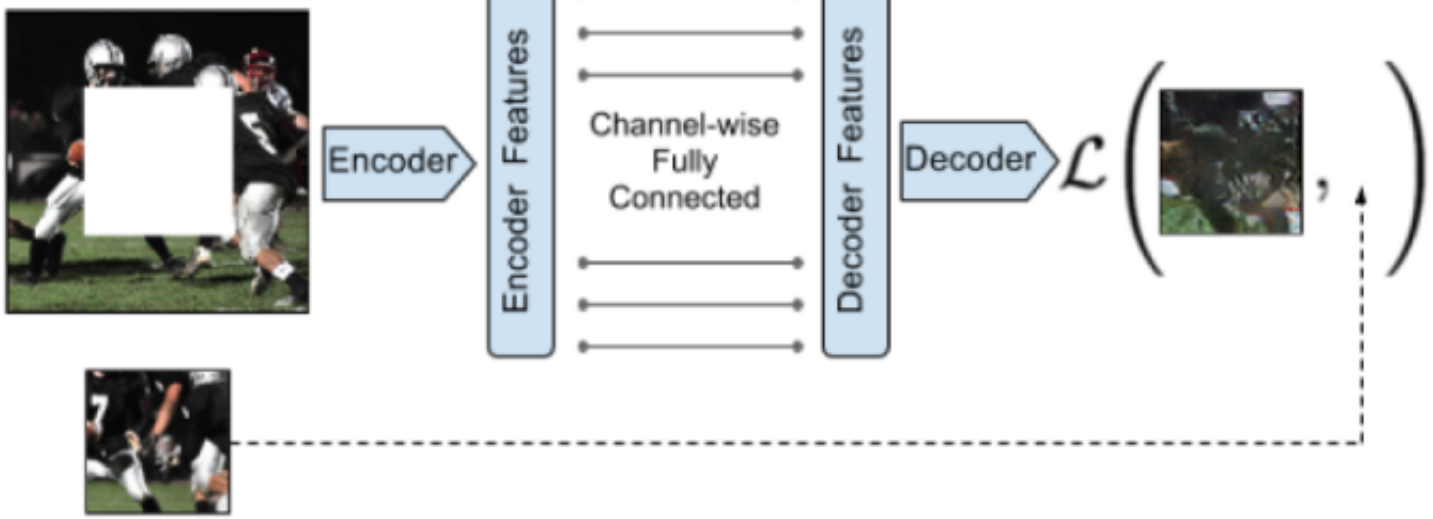
Context Encoder: Feature Learning by Inpainting (Pathak et al)

- A CNN, called Context Encoders, is trained to generate the contents of an arbitrary image region conditioned on its surroundings.
- Both a standard pixel-wise reconstruction loss, as well as a reconstruction plus an adversarial loss, are tested.
- The Context Encoders can be used for feature learning. The learnt features can be usd pre-training on classification, detection, and segmentation tasks, which is a kind of self-supervised learning.

Context Encoder: Feature Learning by Inpainting



Context Encoder: Feature Learning by Inpainting



Context Encoders for Image Generation

Context Encoder: Feature Learning by Inpainting

- For each ground truth image x, the proposed context encoder F produces an output $F(x)$.
- Let $\hat{M}$ be a binary mask corresponding to the dropped image region with a value of 1 wherever a pixel was dropped and 0 for input pixels.
- The reconstruction loss is a **normalized masked L2 distance:**

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2,$$

- where $\odot$ is the element-wise product operation.
- L1 and L2 losses have no significant difference, it often **fails to capture any high frequency details**, often prefer a **blurry** solution, over highly accurate textures.

Context Encoder: Feature Learning by Inpainting

- The adversarial loss tries to make prediction look real, and has the effect of picking a particular mode from the distribution.
- Only the generator (not the discriminator) is conditioned on context when trained using GAN. The adversarial loss for context encoders, *Ladv*, is:

$$\mathcal{L}_{adv} = \max_{D} \quad \mathbb{E}_{x \in \mathcal{X}}[\log(D(x)) + \log(1 - D(F((1 - \hat{M}) \odot x)))],$$

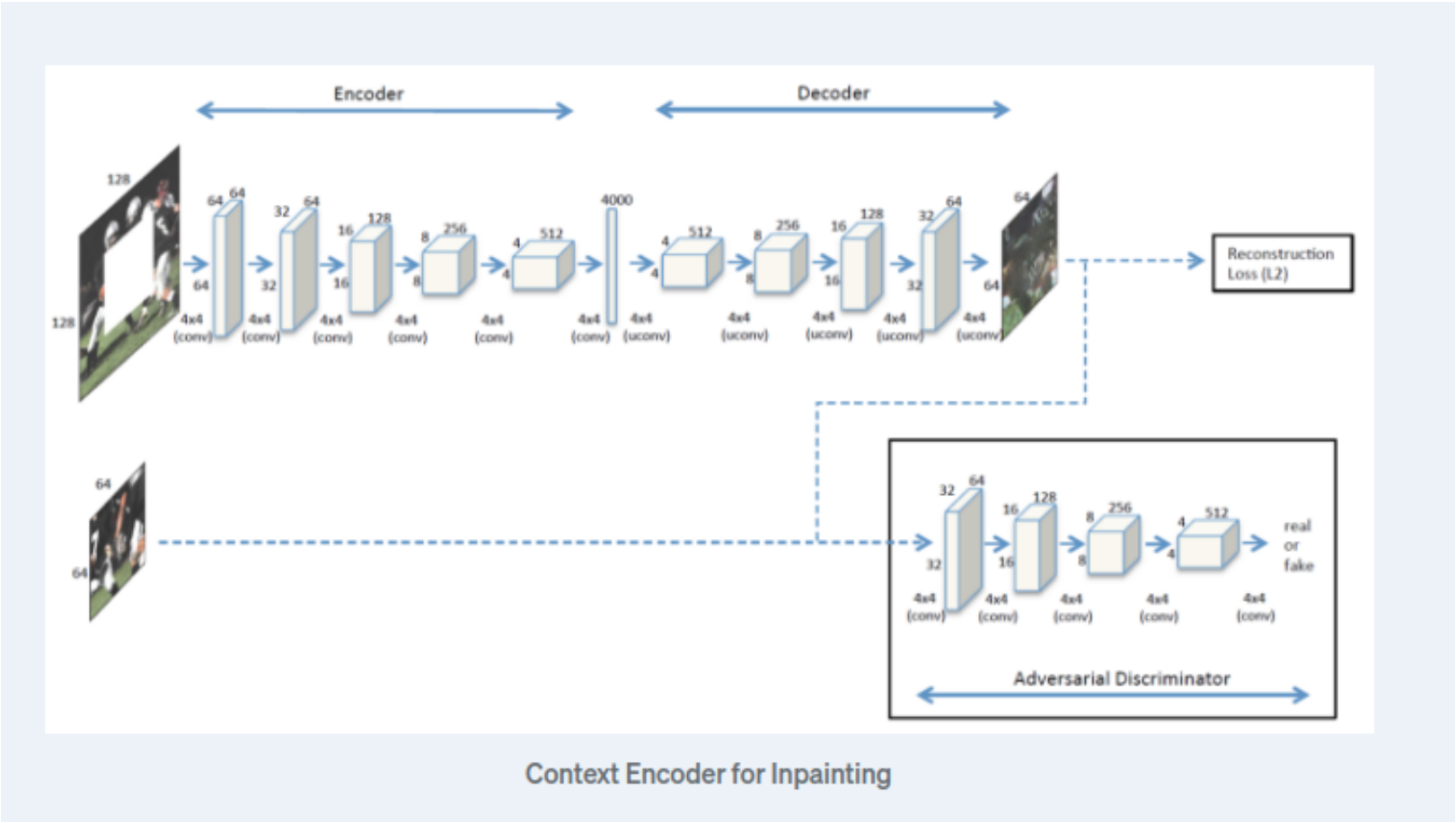
- **Both F and D are optimized jointly using alternating SGD.**

Context Encoder: Feature Learning by Inpainting

- The overall loss function is:

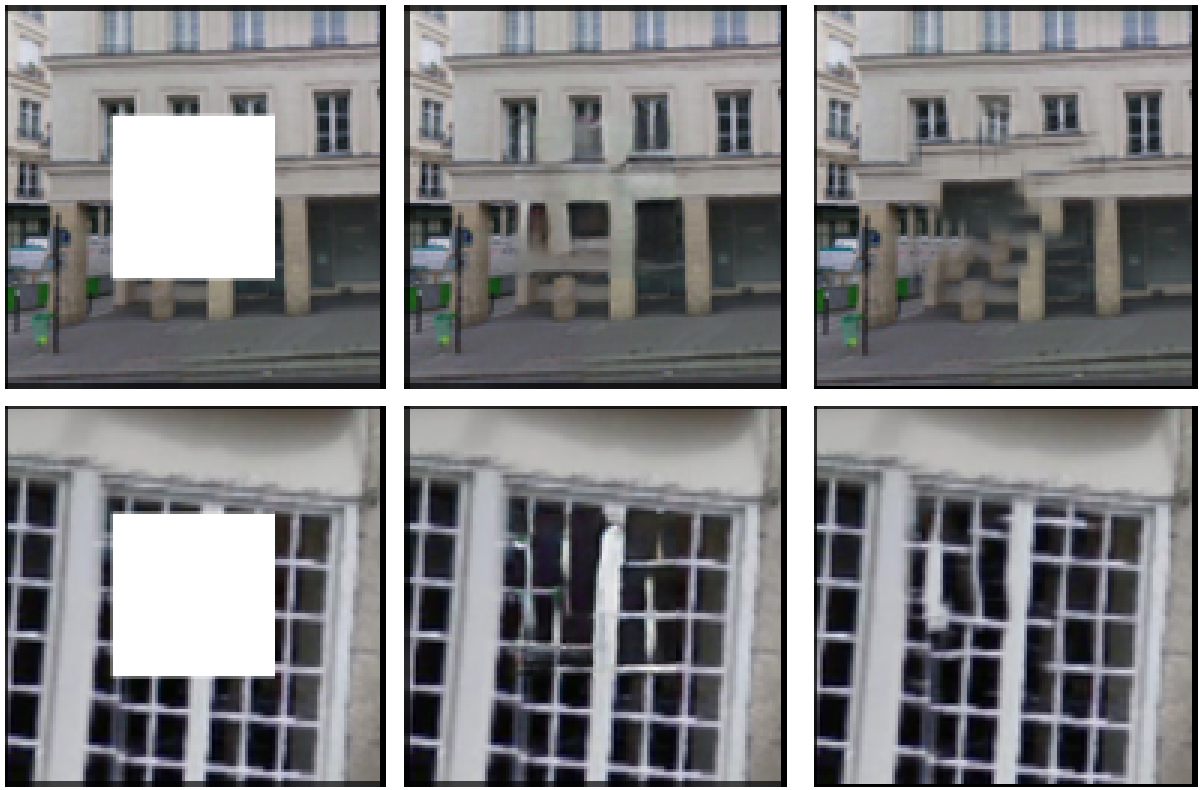
$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}.$$

Context Encoder: Feature Learning by Inpainting

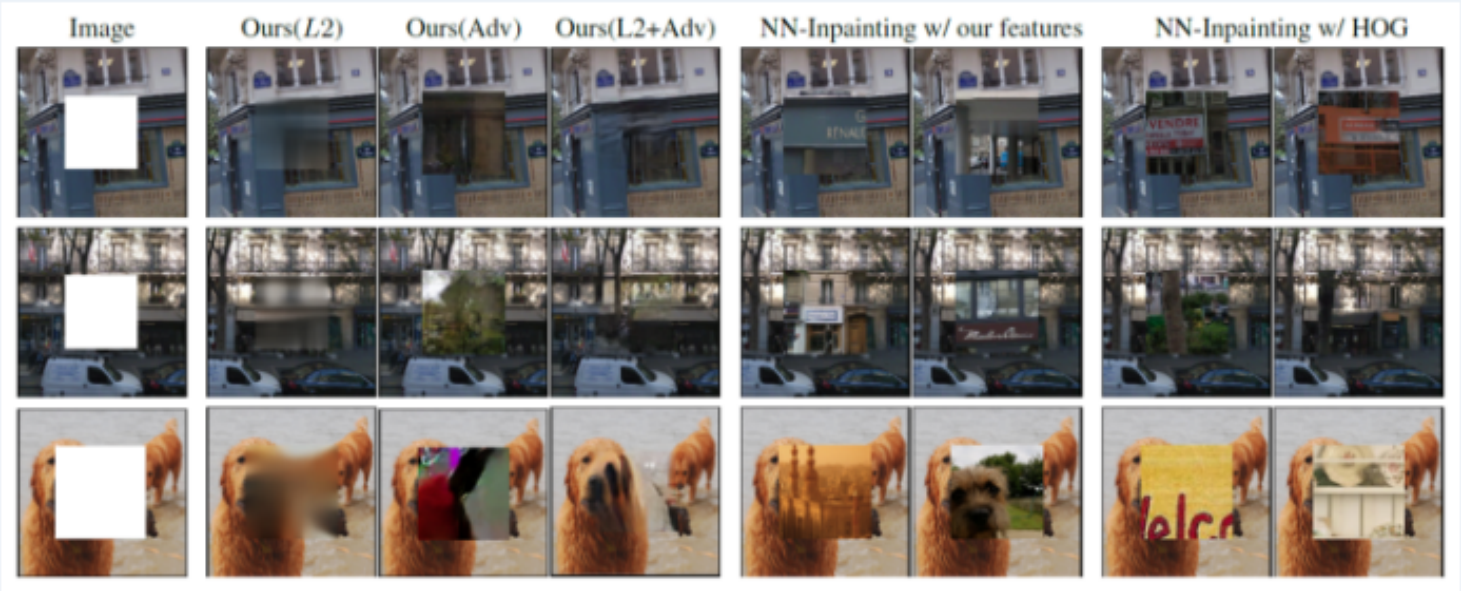


Context Encoder for Inpainting

Context Encoder: Feature Learning by Inpainting

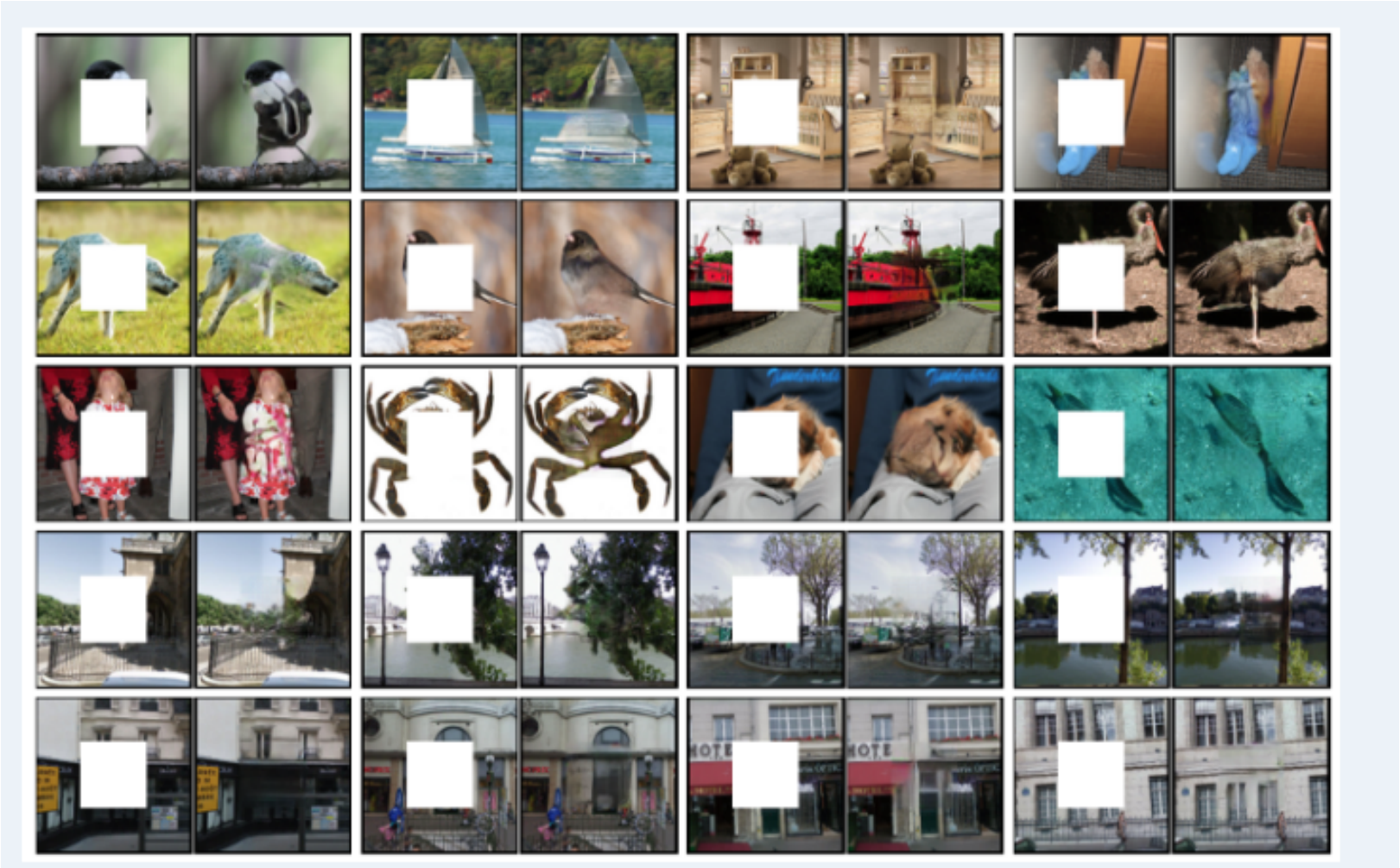


Context Encoder: Feature Learning by Inpainting



Semantic Inpainting using different methods on held-out images

Context Encoder: Feature Learning by Inpainting



Applications of Adversarial loss

- Image generation: colonization, super resolution.....
- Domain Adaptation
- Fair learning
- Active learning
- ....

Next Class

- Self supervised learning
- Zero shot learning
- One shot learning
- Few shot learning
- Domain adaptation
- Active learning
- Bayesian learning
- Fair learning